Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Кафедра приборостроения и информационно-измерительных
технологий

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ
«ПРЕОБРАЗОВАНИЕ ИЗМЕРИТЕЛЬНЫХ СИГНАЛОВ»

Составитель
В.П. ЛЕГАЕВ

Владимир 2009

УДК 621.002.56 (076)
ББК 34.54
М54

Рецензент
Кандидат технических наук, доцент
зав. кафедрой технико-технологических дисциплин»
Владимирского государственного гуманитарного университета
Л.Н. Шарыгин

Печатается по решению редакционного совета
Владимирского государственного университета

М54 **Методические** указания к лабораторным работам по дисциплине «Преобразование измерительных сигналов» / сост. В.П. Легаев ; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2009. – 87 с.

Приведены лабораторные работы по таким разделам курса «Теоретические основы информационной техники», как кодирование информации, модуляция носителей информации, дискретизация информации.

Предназначены для проведения лабораторных работ по дисциплине «Преобразование измерительных сигналов» для студентов дневного и вечернего отделений, обучающихся по специальности 200101 – «Приборостроение».

Ил. 64. Табл. 4. Библиогр.: 6 назв.

УДК 621.002.56 (076)
ББК 34.54

Лабораторная работа № 1
ИССЛЕДОВАНИЕ ПРИНЦИПОВ ДЕМОДУЛЯЦИИ СИГНАЛОВ НА ПРИМЕРЕ ПРОГРАММЫ *lr0* ДЛЯ *MATLAB*

Цель работы: изучение демодуляции различных сигналов с практическим исследованием их характеристик.

Теоретическая часть

Сигналы от измерительных датчиков и любых других источников сигналов передаются по линиям связи к измерительным приборам и в измерительно-вычислительные центры регистрации и обработки данных. Как правило, информационные сигналы являются низкочастотными и ограниченными по ширине спектра в отличие от широкополосных высокочастотных каналов связи, рассчитанных на передачу сигналов от множества источников одновременно с частотным разделением каналов. Перенос спектра сигналов из низкочастотной области в выделенную для их передачи область высоких частот канала связи выполняется операцией модуляции.

Допустим, что низкочастотный сигнал, подлежащий передаче по какому-либо каналу связи или радиоканалу, задается функцией *s(t)*. В канале связи выделяется для передачи данного сигнала определенный высокочастотный диапазон. На входе канала связи в специальном передающем устройстве формируется вспомогательный, как правило, непрерывный во времени периодический высокочастотный сигнал $u(t) = f(t; a_1, a_2, \ldots a_m)$. Совокупность параметров a_i определяет форму вспомогательного сигнала. Если на один из этих параметров перенести сигнал *s(t)*, т.е. сделать его значение пропорционально зависимым от значения *s(t)* во времени (или по любой другой независимой переменной), то форма сигнала *u(t)* приобретает новое свойство. Она несет информацию, тождественную информации в сигнале *s(t)*. Именно поэтому такой сигнал называют несущим сигналом, несущим колебанием или

просто несущей *(carrier),* а физический процесс переноса информации на параметры несущего сигнала – его модуляцией *(modulation).* Исходный информационный сигнал называют модулирующим *(modulating signal),* результат модуляции – модулированным сигналом *(modulated signal).* Обратную операцию выделения модулирующего сигнала из модулированного колебания называют демодуляцией *(demodulation).*

В качестве несущих сигналов обычно используются гармонические колебания

$$u(t) = U \cdot \cos(\omega t + \varphi),$$

которые имеют три свободных параметра: U, ω и φ. В зависимости от того, на какой из данных параметров переносится информация, различают амплитудную, частотную или фазовую модуляцию несущего сигнала. Частотная и фазовая модуляции тесно взаимосвязаны, поскольку изменяют аргумент функции косинуса, и часто их объединяют под общим названием угловая модуляция *(angle modulation).* В каналах передачи цифровой информации получила также распространение квадратурная модуляция, при которой одновременно изменяются амплитуда и фаза несущих колебаний.

Демодуляция АМ-сигнала

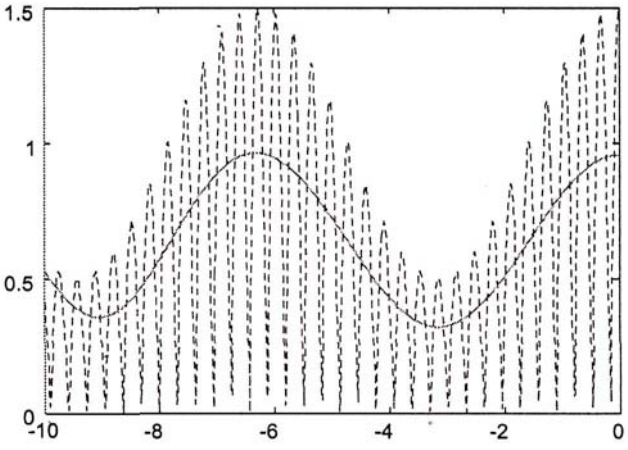


Рис. 1. Двухполупериодное детектирование АМ-сигнала: однополярные импульсы (пунктирная линия) и результат их сглаживания (сплошная линия)

Демодуляция амплитудно-модулированного (АМ)-сигнала может быть выполнена несколькими способами. Простейший путь – имитировать работу аналогового двухполупериодного детектора. Мы вычисляем модуль входного АМ-сигнала, а затем сглаживаем получившиеся однополярные косинусоидальные импульсы, пропуская их через фильтр низких частот (ФНЧ) (рис. 1).

Данный способ, очевидно, не будет работать правильно в случае перемодуляции.

Следующий метод – так называемое синхронное детектирование, суть которого состоит в умножении частоты сигнала на опорное колебание с несущей частотой (рис. 2).

$$y(t) = s_{AM}(t)\cos(\omega_0 t + \varphi_0) = A(t)\cos^2(\omega_0 t + \varphi_0) = \frac{1}{2}A(t) + \frac{1}{2}A(t)\cos(2\omega_0 t + 2\varphi_0). \quad (1)$$

Результат умножения содержит два слагаемых. Первое – это искомая амплитудная функция, второе – АМ-сигнал с несущей частотой $2\omega_0$. Этот высокочастотный сигнал легко удаляется путем пропускания сигнала через ФНЧ.

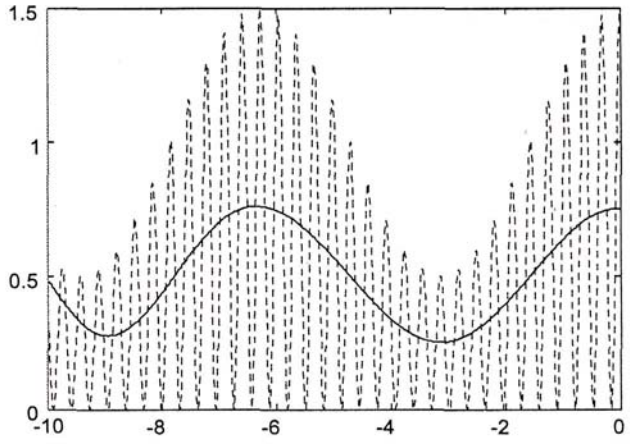


Рис. 2. Синхронное детектирование АМ-сигнала: результат умножения на опорное колебание (пунктирная линия) и выделенный низкочастотный сигнал (сплошная линия)

Однако в данном случае необходимо очень точное совпадение начальных фаз и частот опорного колебания демодулятора и несущего колебания АМ-сигнала. При совпадении частот, но несовпадении начальных фаз выходной низкочастотный сигнал оказывается умноженным на косинус фазовой ошибки

$$y(t) = s_{AM}(t)\cos(\omega_0 t + \varphi_0) = A(t)\cos(\omega_0 t + \varphi_0)\cos(\omega_0 t + \varphi_0) =$$

$$= \frac{1}{2}A(t)\cos(\varphi - \varphi_0) + \frac{1}{2}A(t)\cos(2\omega_0 t + \varphi_0 + \varphi).$$

Таким образом, при наличии фазовой ошибки уровень полезного сигнала на выходе демодулятора падает, а при ошибке, равной 90°, становится равным нулю.

При наличии частотного сдвига между несущим и опорным колебаниями ситуация становится еще хуже – выходной низкочастотный

сигнал оказывается умноженным на гармоническое колебание с разностной частотой

$$y(t) = s_{AM}(t)\cos((\omega_0 + \Delta\omega)t) = A(t)\cos(\omega_0 t + \varphi_0)\cos((\omega_0 + \Delta\omega)t) =$$

$$= \frac{1}{2}A(t)\cos(\Delta\omega t - \varphi_0) + \frac{1}{2}A(t)\cos((2\omega_0 + \Delta\omega)t + \varphi_0).$$

В результате выходной сигнал будет пульсировать с частотой $\Delta\omega$. Это явление называется биениями *(beat)*, а разность частот $\Delta\omega$ - частотой биений *(beat frequency)*.

Для поддержания частотной и фазовой синхронизации между несущим и опорным колебаниями используются следящие системы фазовой автоподстройки частоты (ФАПЧ).

Достоинством синхронного детектирования является то, что оно позволяет правильно демодулировать сигнал даже в случае перемодуляции.

Демодуляция АМ-сигнала с подавленной несущей

Как было сказано в п. «Демодуляция АМ-сигнала», формула (1) справедлива как для однополярной, так и для знакопеременной амплитудной функции *A(t)*. Поэтому демодуляция АМ-сигнала с подавленной несущей может выполняться путем синхронного детектирования. При этом сохраняет силу все сказанное о необходимости точного соответствия частот и начальных фаз несущего и опорного колебания.

Для облегчения правильного восстановления несущей иногда применяют следующий прием. На передающей стороне несущее колебание подавляется не полностью. Его «остаток» с небольшой амплитудой (он называется пилот-сигналом) используют для синхронизации частоты и фазы несущего колебания на приемной стороне.

Демодуляция однополосного сигнала

Несмотря на то что визуально однополосный сигнал сильно отличается от обычного АМ-сигнала, его демодуляция возможна тем же методом синхронного детектирования – путем умножения на опорное колебание

$$y(t) = s_{ssb}(t)\cos\omega_0 t = (x(t)\cos\omega_0 t \pm x_\perp(t)\sin\omega_0 t)\cos\omega_0 t =$$

$$= \frac{1}{2}x(t) + \frac{1}{2}x(t)\cos 2\omega_0 t \pm \frac{1}{2}x_\perp(t)\sin 2\omega_0 t.$$

Результат умножения содержит два слагаемых. Первое – это модулирующий сигнал, второе – однополосный сигнал на удвоенной несущей $2\omega_0$. Остается лишь выделить модулирующий сигнал с помощью ФНЧ (снизу рис. 3).

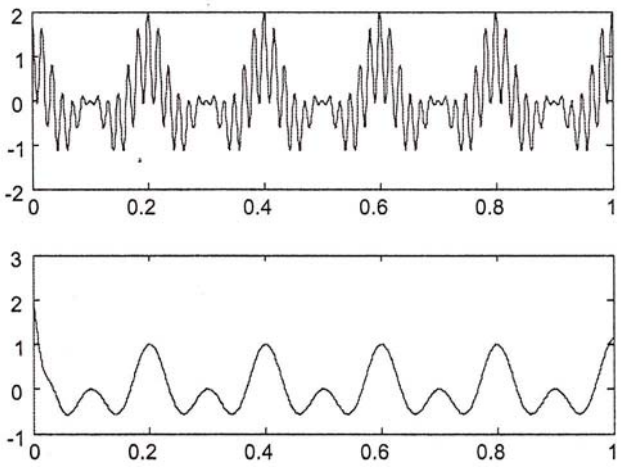


Рис. 3. Демодуляция однополосного сигнала. Сверху – результат умножения на несущее колебание, снизу – отфильтрованный демодулированный сигнал

Эффекты, возникающие при наличии частотного или фазового сдвига у опорного колебания, в случае однополосного сигнала проявляются не так, как при демодуляции АМ-сигнала. Сначала рассмотрим ситуацию, когда есть только фазовый сдвиг, а ошибка по частоте отсутствует

$$y(t) = s_{ssb}(t)\cos(\omega_0 t + \varphi) = (x(t)\cos\omega_0 t \pm x_\perp(t)\sin\omega_0 t)\cos(\omega_0 t + \varphi) =$$
$$= x(t)\cos\omega_0 t\cos(\omega_0 t + \varphi) \pm x_\perp(t)\sin\omega_0 t\cos(\omega_0 t + \varphi) =$$
$$= \frac{1}{2}(x(t)\cos\varphi \pm x_\perp(t)\sin\varphi) + \frac{1}{2}(x(t)\cos(2\omega_0 t + \varphi) \pm x_\perp(t)\sin(2\omega_0 t + \varphi)).$$

Низкочастотная составляющая в этом случае представляет собой линейную комбинацию модулирующего сигнала *x(t)* и его квадратурного дополнения $x_\perp(t)$. Со спектральной точки зрения это означает фазовый сдвиг всех частотных составляющих сигнала на φ. Форма сигнала, разумеется, при этом искажается. Приемлемы ли такие искажения, зависит от характера передаваемого сигнала. Если однополосная модуляция используется для передачи звукового сигнала, фазовые

искажения несущественны, поскольку ухо человека нечувствительно к фазовым соотношениям в акустическом сигнале

$$y(t) = s_{ssb}(t)\cos((\omega_0 + \Delta\omega)t) = \frac{1}{2}(x(t)\cos(\Delta\omega t) \pm x_{\perp}(t)\sin(\Delta\omega t)) +$$

$$+\frac{1}{2}(x(t)\cos((2\omega_0 + \Delta\omega)t \pm x_{\perp}(t)\sin(2\omega_0 + \Delta\omega)t)).$$

Низкочастотная составляющая здесь фактически представляет собой однополосный сигнал с несущей частотой $\Delta\omega$. С частотной точки зрения это означает сдвиг спектра модулирующего сигнала на $\Delta\omega$. Это, разумеется, значительно более серьезные искажения, чем в предыдущем случае. Скажем, при передаче же речевого сигнала (а профессиональная радиосвязь – это основная сфера применения однополосной модуляции) сдвиг спектра приводит к искажению тембра голоса, но разборчивость речи сохраняется при величине сдвига до нескольких десятков, а для опытного оператора – до нескольких сотен герц.

Демодуляция УМ (угловая)

Как и в случае АМ, демодуляция УМ-сигнала может выполняться различными способами. Наиболее радикальный подход – вычислить аналитический сигнал и выделить его фазовую функцию. Дальнейшие действия зависят от вида угловой модуляции. Для демодуляции фазово-модулированного (ФМ) сигнала из фазовой функции вычитается линейное слагаемое $\omega_0 t$, соответствующее немодулированной несущей. В случае частотной модуляции (ЧМ) фазовая функця дифференцируется, а из результата вычитается константа ω_0 (рис. 4).

Недостатком данного метода считается обработка всего входного сигнала сразу с использованием преобразования Гильберта. Для реализации демодуляции в реальном масштабе времени необходима последовательная обработка поступающих отсчетов входного сигнала. Это можно реализовать, используя для получения аналитического сигнала приближенную гильбертовскую фильтрацию во временной области.

Еще одной альтернативой, пригодной для реализации в реальном масштабе времени, является квадратурная обработка. При этом входной сигнал умножается на два опорных колебания, сдвиг по фазе между которыми составляет 90°

$$y_I = s_{УМ}(t)\cos\omega_0 t = A\cos(\omega_0 t + \varphi(t))\cos\omega_0 t = \frac{A}{2}\cos\varphi(t) + \frac{A}{2}\cos(2\omega_0 t + \varphi(t)),$$

$$y_q(t) = s_{УМ}(t)\sin\omega_0 t = A\cos(\omega_0 t + \varphi(t))\sin\omega_0 t = \frac{A}{2}\sin\varphi(t) + \frac{A}{2}\sin(2\omega_0 t + \varphi(t)).$$

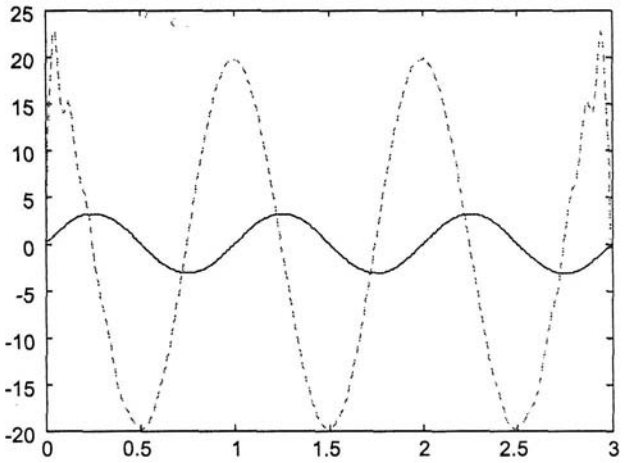


Рис. 4. Результаты фазовой (сплошная линия) и частотной (пунктирная линия) демодуляции сигнала с гармонической УМ

Каждый из результатов умножения содержит два слагаемых. Одно из них – низкочастотное (косинус или синус начальной фазы), другое – высокочастотное (УМ-сигнал с несущей частотой $2\omega_0$). Низкочастотные составляющие выделяются с помощью ФНЧ

$$y'_I(t) = \frac{A}{2}\cos\varphi(t),$$

$$y'_Q(t) = -\frac{A}{2}\sin\varphi(t).$$

Дальнейшие действия, так же как и раньше, зависят от вида угловой модуляции. Для демодуляции ФМ нам необходимо вычислить фазу полученной пары квадратурных составляющих

$$x_{ФМ}(t) = -\arg(y'_I + jy'_Q(t)) = -\arg\left(\frac{A}{2}\cos\varphi(t) - j\frac{A}{2}\sin\varphi(t)\right) =$$

$$= -\arg\left(\frac{A}{2}\exp(-j\varphi(t))\right) = \varphi(t).$$

Для демодуляции ЧМ полученную фазовую функцию необходимо продифференцировать

$$x_{ЧМ}(t)=\frac{dx_{ФМ}}{dt}=-\frac{d}{dt}\arg\left(y'_I(t)+jy'_Q(t)\right)=-\frac{d}{dt}arctg\frac{y'_Q(t)}{y'_I(t)}=\frac{\frac{dy'_I}{dt}y'_Q(t)-\frac{dy'_Q}{dt}y'_I(t)}{y'^2_I(t)+y'^2_Q(t)}.$$

Структурная схема получившегося демодулятора показана на рис. 5.

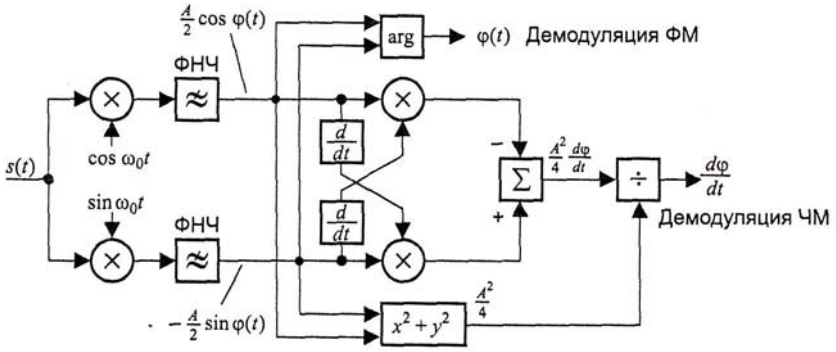


Рис. 5. Квадратурная обработка сигнала с угловой модуляцией

Достоинство данной схемы в том, что при высокой несущей частоте входные блоки (генератор опорного колебания и умножители) могут быть выполнены в аналоговом виде.

Еще одним способом демодуляции сигналов с УМ является использование следящих систем ФАПЧ.

На рис. 6 изображено рабочее окно программы *lr0*. Выносными линиями обозначены основные элементы интерфейса.

Форма исходного сигнала выбирается из соответствующего списка, состоящего из подпунктов: *speech* (речь), *sine* (синусоидальная форма сигнала), *square* (квадратная форма сигнала) и *triangle* (треугольная форма). В поле *Fc* задается частота передаваемого сигнала, а в поле *Fc* – несущая частота.

Выбор методов демодуляции осуществляется точечным переключателем. AM соответствует амплитудной, FM – частотной, а PM – фазовой демодуляции.

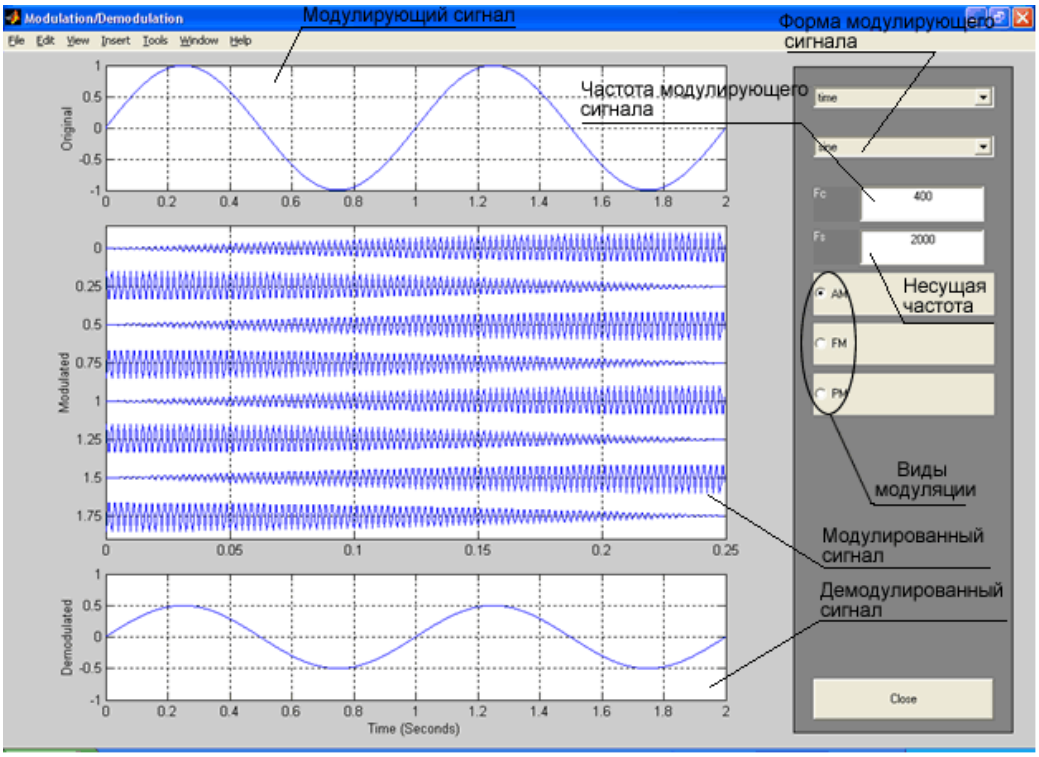


Рис. 6. Рабочее окно программы *lr0*

Порядок выполнения работы

Ознакомиться с основными принципами модуляции сигналов и интерфейсом программы *lr0*.

1. Запустить программу *Matlab* с помощью ярлыка на рабочем столе или из меню Пуск-Программы-*Matlab Realese N-matlab*. Дождавшись загрузки приложения в окне “command window”, набрать lr0 и нажать enter.
2. В поле «форма модулирующего сигнала» выбрать sine.
3. Установить значение модулирующей частоты 300 Гц, а несущей 1500 Гц.
4. Зарисовать полученные графики демодулированного сигнала для AM, FM и PM демодуляций.
5. Построить графики для частот *Fc*=600 Гц *Fs*=3000 Гц и *Fc*=1500 Гц *Fs*=17000 Гц.
6. Построить графики для квадратной и треугольной форм сигнала со значениями *Fc* и *Fs,* указанными для sine.
7. Сделать выводы по графикам.

Содержание отчета

Отчет должен содержать

1. Графики демодулированного сигнала для AM, FM и PM демодуляций.
2. Графики для частот *Fc*=600 Гц *Fs*=3000 Гц и *Fc*=1500 Гц *Fs*=17000 Гц.
3. Графики для квадратной и треугольной форм сигнала со значениями *Fc* и *Fs,* указанными для sine.
4. Выводы по полученным графикам.

Контрольные вопросы

1. Назовите известные вам виды сигнала, получаемые от измерительных преобразователей.
2. Перечислите способы демодуляции полученных сигналов.
3. Сделайте выводы.

Лабораторная работа № 2

СОЗДАНИЕ МАССИВОВ СО СЛУЧАЙНЫМИ ЭЛЕМЕНТАМИ

Цель работы: научиться создавать массивы со случайными элементами.

Теоретическая часть

Ознакомление с синтаксисом команд.

p = randperm(n) — возвращает случайные перестановки целых чисел 1:*n* в векторе-строке. Пример:

» randperm(6)

ans = 243651

Функция *rand* генерирует массивы случайных чисел, значения элементов которых равномерно распределены в промежутке (0, 1):

rand(n) — возвращает матрицу размером *nxn*. Если *n* — не скаляр, то появится сообщение об ошибке;

rand(m.n) или *rand([m n])* — возвращает матрицу размером *mxn*;

rand(m.n,p....) или *rand([m n p...])* — возвращает многомерный массив;

rand(size(A)) — возвращает массив того же размера и размерности, что и A, с элементами, распределенными по равномерному закону;

rand (без аргументов) — возвращает одно случайное число, которое изменяется при каждом последующем вызове и имеет равномерный закон распределения;

rand(' state') — возвращает вектор с 35 элементами, содержащий текущее состояние генератора случайных чисел с равномерным распределением. Для изменения состояния генератора можно применять следующие формы этой функции:

rand('state' .s) — устанавливает состояние в *s*;

rand('state' ,0) — сбрасывает генератор в начальное состояние;

rand('state'. j) — для целых *j* устанавливает генератор в *j*-е состояние;

*rand('state' ,sum(100*clock))* — каждый раз сбрасывает генератор в состояние, зависящее от времени.

Порядок выполнения работы

1. Используя перечень команд, генерировать случайные числа, массивы разной размерности. Пример:

» *Y=rand*(4,3)

Y=

0.9501	0.8913	0.8214
0.2311	0.7621	0.4447
0.6068	0.4565	0.6154
0.4860	0.0185	0.7919

2. Графическое изображение массива со случайными элементами.

Проверить равномерность распределения случайных чисел можно, построив большое число точек на плоскости со случайными координатами. Это делается с помощью следующих команд:

» *X=rand(1000,1);*

» *Y=rand(1000,1);*

» *plot(X,Y,'.')*

Полученный при этом график показан на рис. 1. Нетрудно заметить, что точки довольно равномерно распределены на плоскости, так что нет оснований не доверять заданному закону распределения координат точек (рис. 7).

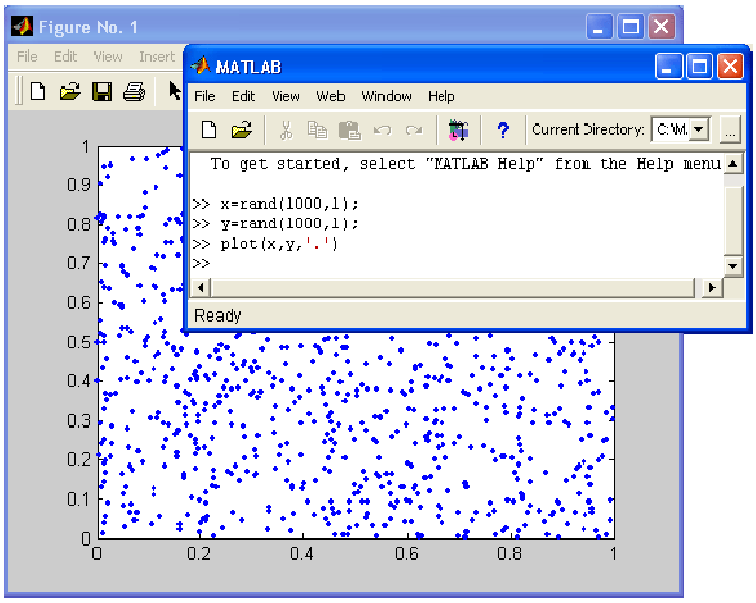


Рис. 7. Случайные точки с равномерным распределением координат на плоскости

3. Исследование массивов со случайными элементами по нормальному закону.

Функция *randn* генерирует массив со случайными элементами, распределенными по нормальному закону с нулевым математическим ожиданием и среднеквадратическим отклонением, равным 1:

randn(n) — возвращает матрицу размером *nxn*. Если *n* — не скаляр, то появится сообщение об ошибке;

randn(m.n) или *randn([m n])* — возвращает матрицу размером mxn;

randn(m,n,p,...) или *randn([m n p...])* — возвращает массив с элементами, значения которых распределены по нормальному закону;

randn(size(A)) — возвращает массив того же размера, что и *A*, с элементами, распределенными по нормальному закону;

randn (без аргументов) — возвращает одно случайное число, которое изменяется при каждом последующем вызове и имеет нормальное распределение;

randn('state') — возвращает двухэлементный вектор, включающий текущее состояние нормального генератора. Для изменения состояния генератора можно применять следующие формы этой функции:

randn('state',s) — устанавливает состояние в *s*;

randn('state' ,0) — сбрасывает генератор в начальное состояние;

randn('state', j) — для целых *j* устанавливает генератор в *J*-е состояние;

*randn('state', sum(100*clock))* — каждый раз сбрасывает генератор в состояние, зависящее от времени.

4. Используя перечень команд, генерировать случайные числа, массивы с элементами распределёнными по нормальному закону.

Пример:

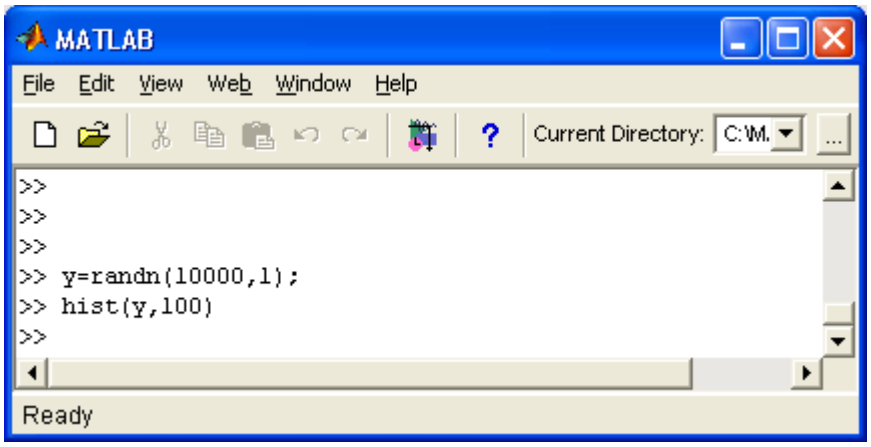
```
>>Y=randn(4.3)
Y =
-0.4326   -1.1465    0.3273
-1.6656    1.1909    0.1746
0.1253     1.1892   -0.1867
0.2877    -0.0376    0.7258
```

5. Графическое представление нормального закона.

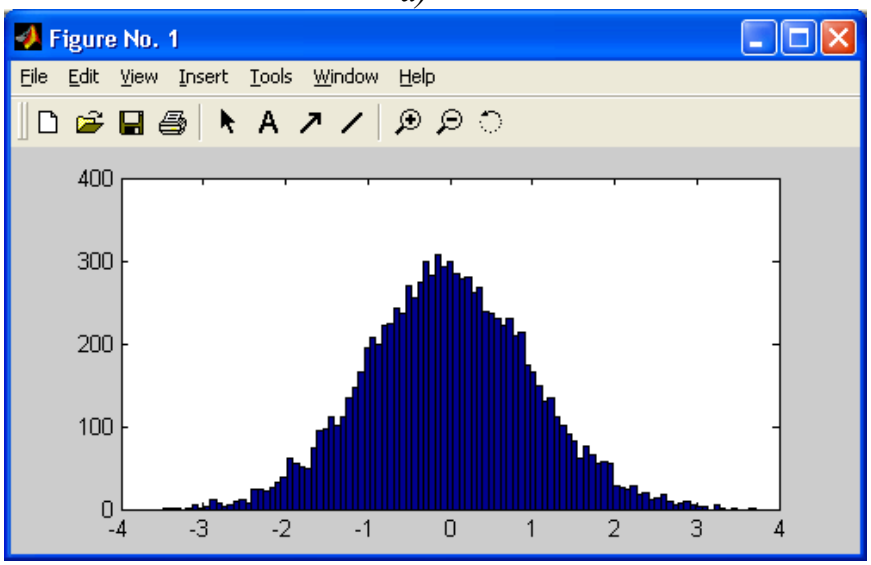
Проверить распределение случайных чисел по нормальному закону можно, построив гистограмму распределения большого количества чисел. Например, следующие команды

```
» Y=randn(10000,1);
```

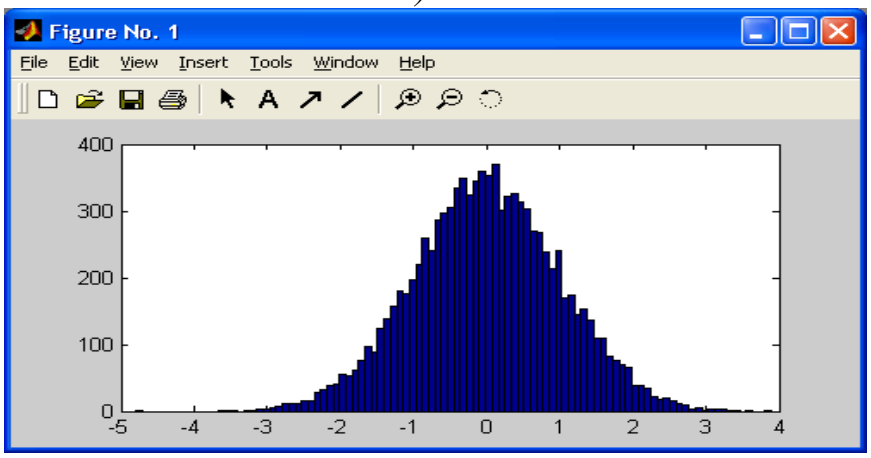
» hist(Y,100) строят гистограмму (рис. 8) из 100 столбцов для 10 000 случайных чисел с нормальным распределением.



а)



б)



в)

Рис. 8. Гистограмма для 10 000 нормально распределенных чисел в 100 интервалах: а – окно исходных данных для расчета; б – равномерное распределение точек; в – нормальное распределение точек

Из рисунка видно, что огибающая гистограммы действительно близка к нормальному закону распределения.

В пакете расширения *Statistics Toolbox* можно найти множество статистических функций, в том числе для генерации случайных чисел с различными законами распределения и определения их статистических характеристик.

Содержание отчета

Отчет должен отразить

1. Случайные точки с равномерным распределением координат на плоскости.
2. Гистограмму для 10 000 нормально распределенных чисел в 100 интервалах.

Контрольные вопросы

1. Каким образом можно задать массив случайных чисел размерностью 5*x*6?
2. Возможно ли задать многомерный массив?
3. Как можно задать одно случайное число?
4. Каким образом можно изменять текущее состояние генератора текущих чисел?

Лабораторная работа № 3

РАСЧЕТ ЦИФРОВЫХ ФИЛЬТРОВ В СРЕДЕ MATLAB

Цель работы: научиться выполнять расчет цифровых фильтров в среде *MATLAB*.

Теоретическая часть

1. Цифровые фильтры

Цифровые фильтры предназначены для обработки (фильтрации) сигналов, представленных в виде временных рядов. По виду импульсной характеристики цифровые фильтры принято делить на фильтры с конечной импульсной характеристикой (КИХ-фильтры) и с бесконечной импульсной характеристикой (БИХ-фильтры).

Обычно нерекурсивные цифровые фильтры имеют КИХ, рекурсивные – БИХ. Первые в отличие от вторых могут иметь линейную фазочастотную характеристику (ФЧХ), но требуют больших вычислительных затрат (ВЗ). Рекурсивные конечно-импульсные характеристики фильтров (РКИХФ) также могут иметь линейную ФЧХ, но при гораздо меньших ВЗ по сравнению с нерекурсивным фильтрами. До недавнего времени были известны РКИХФ с примитивными импульсными характеристиками (ИХ): прямоугольной или в виде отрезка комплексной экспоненты с прямоугольной огибающей.

Цифровой фильтр полностью описывается его импульсной характеристикой. Импульсная характеристика – это реакция фильтра на единичный импульс, поданный на его вход. Другая характеристика, используемая при изучении и проектировании цифровых фильтров ЦФ, называется переходной. Она показывает, для чего предназначен фильтр с разными частотными составляющими входного сигнала, как он изменяет спектр сигнала. Обе эти характеристики не являются независимыми, они связаны между собой преобразованием Фурье. Зная одну из них, с помощью прямого или обратного преобразования Фурье можно получить другую. Переходная характеристика показывает, как фильтр изменяет амплитуду и фазу частотных составляющих входного сигнала. Так как переходная характеристика – это комплексная функция частоты и изобразить ее на бумаге довольно трудно, а может быть и бессмысленно, то принято рисовать отдельно зависимость амплитуды от частоты – амплитудно-частотная характеристика (АЧХ) и зависимость сдвига фазы от частоты – фазочастотная характеристика (ФЧХ). Если хотя бы один коэффицент А не равен нулю, то мы получаем рекурентную формулу для фильтра, когда любое значение входного ряда оказывает влияние на результат. При этом импульсная характеристика будет иметь бесконечное число пусть и очень маленьких, но ненулевых значений. Отсюда возникли и названия фильтров. Примерами БИХ-фильтров служат экспоненциальное среднее и все использующие его индикаторы, например индикатор *TRIX*.

Количество ненулевых коэффициентов A принято называть порядком фильтра. Чем больше порядок фильтра, тем более эффективный фильтр можно построить. Например, формула экспоненциального сглаживания содержит один ненулевой коэффициент A и, следовательно, этот индикатор является БИХ-фильтром 1-го порядка. Индикатор *TRIX* содержит тройное экспоненциальное сглаживание и, следовательно, три ненулевых коэффициента A, т.е. он является БИХ-фильтром 3-го порядка.

Цифровые фильтры также принято делить по виду АЧХ. Амплитудно-частотная характеристика может иметь самый разный вид. Принято выделять фильтры низких частот (ФНЧ), фильтры высоких частот (ФВЧ), полосовые фильтры (ПФ), заграждающие фильтры (ЗФ), дифференцирующие фильтры и некоторые другие.

В трейдинге в основном используются ФНЧ – это все скользящие средние и дифференцирующие в комбинации с ФНЧ – это большинство осцилляторов.

2. Расчет цифровых фильтров в командном окне

Расчет коэффициентов a_k нерекурсивного фильтра с помощью функции fir1.

Функция *fir1* реализует вычисления по методу обратного преобразования Фурье с использованием окон

a=fir1(n,Wn,'ftype',window,'normalization')

Здесь:

n – порядок фильтра – целое четное число (количество коэффициентов фильтра равно *n*+1);

Wn – относительная частота среза (по отношению к частоте Найквиста, равной половине частоты дискретизации *Fd*) – число в диапазоне (0,1); является вектором из двух чисел, если фильтр полосовой или режекторный;

'ftype' – тип фильтра (если отсутствует или *'low'*– ФНЧ; *'high'* – ФВЧ; *'bandpass'* или отсутствует – полосовой; '*stop*' – режекторный;

Примечание: в *Matlab* 5.3 явное указание типа фильтра '*low'* приводит к отказу от вычислений – в этом случае параметр *'ftype'* нужно опускать. Может оказаться, что по вине разработчиков данная осо-

бенность присуща и другим версиям *Matlab*. Кроме того, возможно, что подобная особенность свойственна и типу *'bandpass'* (нами не проверялось).

window – вектор-столбец из *n+1* элементов (по умолчанию применяется окно Хэмминга *hamming(n+1));*

'normalization' – нормировка АЧХ и импульсно-полосовая характеристика (ИПХ) (по умолчанию значение *'scale'* – единичное значение АЧХ в центре полосы пропускания; *'noscale'* – нормировка не производится).

Примечание: при нормализации максимальное значение АЧХ в точности равно единице. При отсутствии нормализации из-за эффекта Гиббса максимальное значение АЧХ больше единицы. Этим и объясняется, что рассчитанные коэффициенты фильтра в отсутствие нормализации больше таковых при наличии нормализации (см. приводимый ниже пример).

Пример:

window=rectwin(7) % синтез прямоугольного окна из 7 отсчетов

a=fir1(6,0.5,window) % расчет коэффициентов КИХ-фильтра с нормализацией.

Результат:

a=[-0.1148 0.0000 0.3443 0.5409 0.3443 0.0000 -0.1148].

Сравнивая эти результаты с рассчитанными вручную коэффициентами, нетрудно видеть разницу. Например, ручные расчеты дают $a_0 = 0.5$, тогда как в *Matlab* мы получили $a_0 = 0.5409$. Естественно предположить, что причиной тому проводимая по умолчанию нормировка ИПХ. Проверяем это предположение, задавая в программе значение 'noscale' для параметра нормализации:

window=rectwin(7) % синтез прямоугольного окна из 7 отсчетов

a=fir1(6,0.5,window,'noscale') % расчет коэффициентов КИХ-фильтра без нормализации.

Результат:

a=[-0.1061 0.0000 0.3183 0.5000 0.3183 0.0000 -0.1061].

В приведенном выше примере прямоугольное окно генерировалось с помощью функции *rectwin(n+1)*, где *n* – порядок нерекурсивного фильтра. Следует отметить, что прямоугольное окно можно сгенерировать также с помощью функции *boxcar(n+1).*

3. Расчет фильтра a_k с помощью пакета *sptool*

Для активизации пакета нужно в командном окне набрать команду *sptool*. Затем в появившемся окне в колонке кнопок *"Filters"* нажать кнопку "*New"*.

В появившемся окне *«Filter Designer»:*

задать частоту дискретизации (согласно примеру 1 задаем 100 Гц);

выбрать в позиции *Algoritm* значение *Kaiser Window FIR* (выбираем из 3 вариантов: *Equiripple FIR, Least Square FIR* и *Kaiser Window FIR);*

отключить флажок *Minimum Order*;

задать *Order*=6;

задать *Type=lowpass*;

задать *Passband Fp*=25;

отключить флажок *Autodesigne*;

закрыть окно *Filter Designer*;

в окне *sptool* в колонке *Filters* нажать кнопку *View*;

в появившемся окне *Filter Viewer* наблюдаем графики АЧХ, ФЧХ, ИПХ (ИПХ наблюдаем после активизации соответствующего флажка).

Примечание: мы выбрали в позиции *Algoritm* значение *Kaiser Window FIR*. Кроме данного алгоритма, есть еще два алгоритма: *Equiripple FIR, Least Square FIR.* Из всех этих трех алгоритмов только алгоритм Кайзера реализует метод обратного преобразования Фурье с весовым окном Кайзера. При значении параметра 0 окно Кайзера превращается в обычное прямоугольное окно.

Как показывает эксперимент, рассчитанные таким образом коэффициенты ФНЧ оказываются ненормированными, т.е. в точности равными вычисленным вручную.

Примечание: чтобы узнать значения коэффициентов, нужно:

активизировать график ИПХ, щелкнув по нему мышкой;

активизировать вертикальные маркеры (кнопкой, расположенной под меню);

поместить один из маркеров (всего имеется 2 маркера – 1-й изображается сплошной вертикальной линией, 2-й - пунктирной) напротив нужного отсчета ИПХ;

считать значение отсчета ИПХ в специальном окошке.

На рис. 9 показано окно *Filter Viewer* (в режиме предпечатного просмотра) для данного конкретного случая (маркер 1 установлен против максимального отсчета, равного 0.5, а маркер 2 – против смежного отсчета, равного 0.3183).

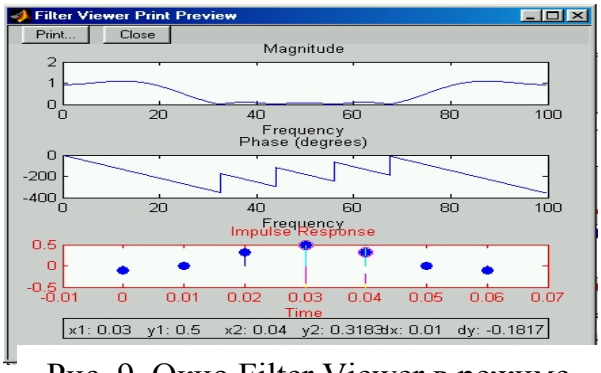


Рис. 9. Окно Filter Viewer в режиме предпечатного просмотра

Пакет *sptool* позволяет моделировать процесс фильтрации с помощью рассчитанного фильтра. Для этого в среду пакета *sptool* нужно импортировать входной сигнал, сгенерированный в рабочем окне программы *Matlab*. Фильтрация происходит после нажатия кнопки *Apply*.

4. Расчет фильтра a_k с помощью пакета *fdatool*

Для активизации пакета нужно в командном окне набрать команду *fdatool*. Затем в появившемся окне в разделе *Designe Filter* задать:

Designe Method: FIR=Window;

Window Specifications: Window=Rectangular;

Filter order: Specify order=6;

частоту дискретизации (*Fs*=100 Гц);

Filter Type=lowpass;

Passband Fc=25;

с помощью кнопок под меню включаем режим просмотра коэффициентов фильтра.

В результате проведенных расчетов убеждаемся, что здесь по умолчанию производится нормирование ИПХ (см. выше *fir1*).

Пакет *fdatool* не позволяет моделировать процесс фильтрации с помощью рассчитанного фильтра. Однако рассчитанные коэффициенты фильтра можно импортировать из среды *fdatool* в среду *Matlab*, где их можно использовать для моделирования фильтрации.

5. Сопоставление способов расчета

Расчеты с помощью пакета *sptool*: достоинство – не нужно помнить синтаксис команд; недостаток – ограниченный набор окон для метода

обратного преобразования Фурье (только окно Кайзера); особенность – вычисляются только ненормированные коэффициенты фильтра.

Расчеты с помощью пакета *fdatool*: достоинство – не нужно помнить синтаксис команд; особенность – вычисляются только нормированные коэффициенты фильтра.

Порядок выполнения работы

1) Ознакомиться со способами расчета цифровых фильтров в среде *MATLAB:*

- в командном окне;
- с помощью пакета *sptool;*
- с помощью пакета *fdatool.*

2) Выбрать из табл. 1 свой вариант параметров синтезируемого фильтра $ФНЧ_1$ – номер варианта соответствует последней цифре вашего номера в списке студентов вашей группы. (Например, студент Иванов находится в списке под номером 17 – значит, его вариант № 7).

Таблица 1

Вариант	Фильтр	N	$f_д$,Гц	$f_с$,Гц
0	$ФНЧ_1$	3	600	50
	$ФНЧ_2$	3	600	70
1	$ФНЧ_1$	4	900	60
	$ФНЧ_2$	4	900	90
2	$ФНЧ_1$	5	1200	80
	$ФНЧ_2$	5	1200	130
3	$ФНЧ_1$	6	1500	120
	$ФНЧ_2$	6	1500	160
4	$ФНЧ_1$	7	1800	140
	$ФНЧ_2$	7	1800	200
5	$ФНЧ_1$	7	2100	175
	$ФНЧ_2$	7	2100	220
6	$ФНЧ_1$	6	2400	200
	$ФНЧ_2$	6	2400	300
7	$ФНЧ_1$	5	2700	225
	$ФНЧ_2$	5	2700	300
8	$ФНЧ_1$	4	3000	250
	$ФНЧ_2$	4	3000	350
9	$ФНЧ_1$	3	3300	260
	$ФНЧ_2$	3	3300	360

3) С помощью программы *Matlab* вычислить коэффициенты нерекурсивного фильтра нижних частот, построить графики импульсного отклика, АЧХ и ФЧХ фильтра.

4) Повторить действия для другого фильтра - $ФНЧ_2$.

5) Объяснить полученные результаты (графики, уравнения фильтров, аналитические выражения частотных характеритик фильтров), сделать соответствующие выводы.

Содержание отчета

Отчет должен содержать

1. Графики, уравнения фильтров.
2. Аналитические выражения частотных характеристик фильтров.
3. Соответствующие выводы.

Контрольные вопросы

1. Назовите способы расчета цифровых фильтров.
2. Объясните полученные результаты эксперимента (графики, уравнения фильтров, аналитические выражения частотных характеристик фильтров).

Лабораторная работа № 4
ФУНКЦИИ ОДНОМЕРНОГО ПРЯМОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

Цель работы: 1) создать трехчастотный сигнал на фоне сильного шума, построить график спектральной плотности полученного сигнала; 2) аппроксимировать полученную функцию; 3) построить график погрешности.

Теоретическая часть

В описанных ниже функциях реализован особый метод быстрого преобразования Фурье — *Fast Fourier Transform* (*FFT*, или БПФ), позволяющий резко уменьшить число арифметических операций в ходе

преобразований. Он особенно эффективен, если число обрабатываемых элементов (отсчетов) составляет $2^{т}$, где т — целое положительное число. Используется следующая функция:

fft(X) — возвращает для вектора *X* дискретное преобразование Фурье, по возможности используя алгоритм быстрого преобразования Фурье. Если *X* — матрица, функция *fft* возвращает преобразование Фурье для каждого столбца матрицы;

fft(X.n) — возвращает *n*-точечное преобразование Фурье. Если длина вектора *X* меньше *n*, то недостающие элементы заполняются нулями. Если длина *X* больше *n*, то лишние элементы удаляются. Когда *X* — матрица, длина столбцов корректируется аналогично;

fft(X,[Ldirn) и fft(X,n,dim) — применяют преобразование Фурье к одной из размерностей массива в зависимости от значения параметра *dim*.

Порядок выполнения работы

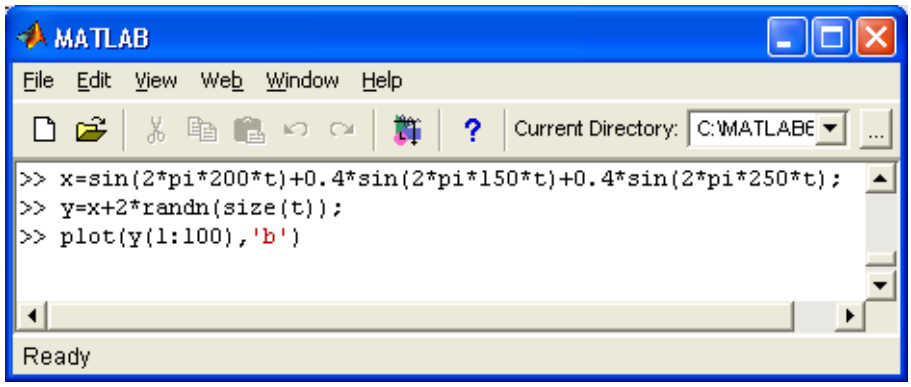
1. Создадим трехчастотный сигнал на фоне сильного шума, создаваемого генератором случайных чисел (рис. 10, *а*).

```
» t=0:0.0005:1;
x=sin(2·pi·200·t)+0.4·sin(2·pi·150·t)+0.4·sin(2·pi·250·t);
y=x+2·randn(size(t));
plot(y(1:100),'b').
```

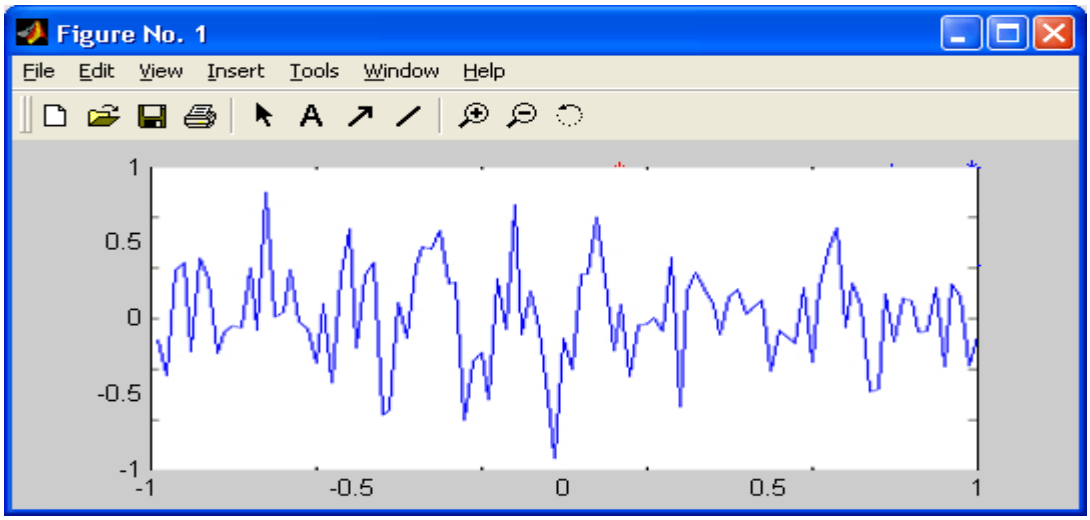
Этот сигнал имеет среднюю частоту 200 рад/с и два боковых сигнала с частотами 150 и 250 рад/с, что соответствует амплитудно-модулированному сигналу с частотой модуляции 50 рад/с и глубиной модуляции 0.8 (амплитуда боковых частот составляет 0.4 от амплитуды центрального сигнала). На рис. 10, *б* показан график этого сигнала (по первым 100 отсчетам из 2000). Нетрудно заметить, что из него не видно, что полезный сигнал — амплитудно-модулированное колебание, настолько оно забито шумами. Теперь построим график спектральной плотности полученного сигнала с помощью прямого преобразования Фурье, по существу переводящего временное пред-

ставление сигнала в частотное. Этот график в области частот до 300 Гц (см. рис. 10, *б*) строится с помощью следующих команд:

» Y=fft(y,1024);
*Pyy=Y.*conj(Y)/1024;*
f=2000(0:150)/1024;*
plot(f,Pyy(1:151)),grid.



а)

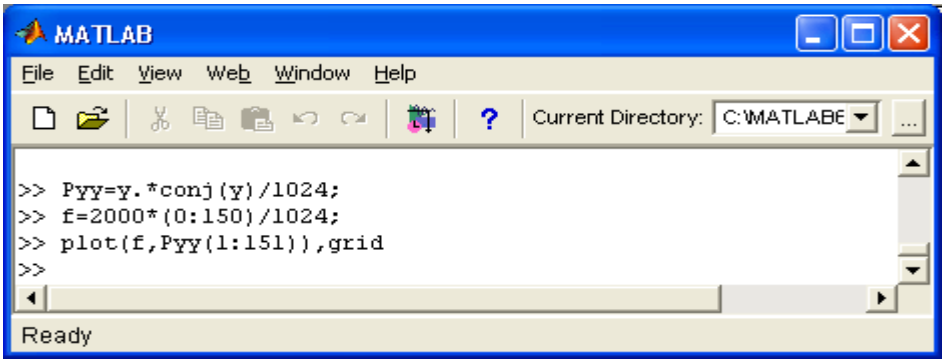


б)

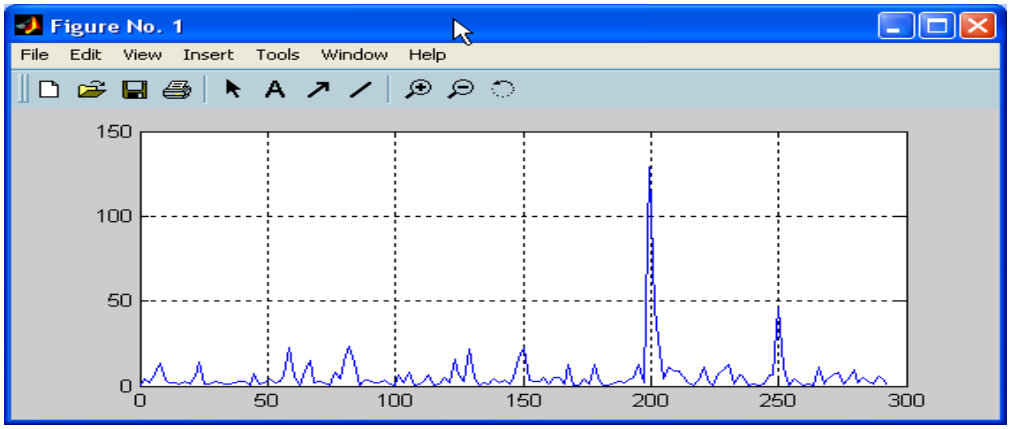
Рис. 10. Форма зашумленного сигнала

График спектральной плотности сигнала, построенный в этом примере (рис. 11, *а*), представлен на рис. 11, *б*. Даже беглого взгляда на рисунок достаточно, чтобы убедиться в том, что спектрограмма сигнала имеет явный пик на средней частоте амплитудно-модулированного сигнала и один боковой пик. Эти две частотные составляющие сигнала явно выделяются на общем шумовом фоне. Таким образом, данный пример наглядно иллюстрирует технику

обнаружения слабых сигналов на фоне шумов, лежащую в основе работы радиоприемных устройств.



а)



б)

Рис. 11. График спектральной плотности сигнала

2. Обработка данных в графическом окне. Средства обработки данных в графическом окне

Решение большинства задач интерполяции и аппроксимации функций и табличных данных обычно сопровождается их визуализацией. Она, как правило, заключается в построении узловых точек функции (или табличных данных) и в построении функции аппроксимации или интерполяции. Для простых видов аппроксимации, например полиномиальной, желательно нанесение на график формулы, полученной для аппроксимации.

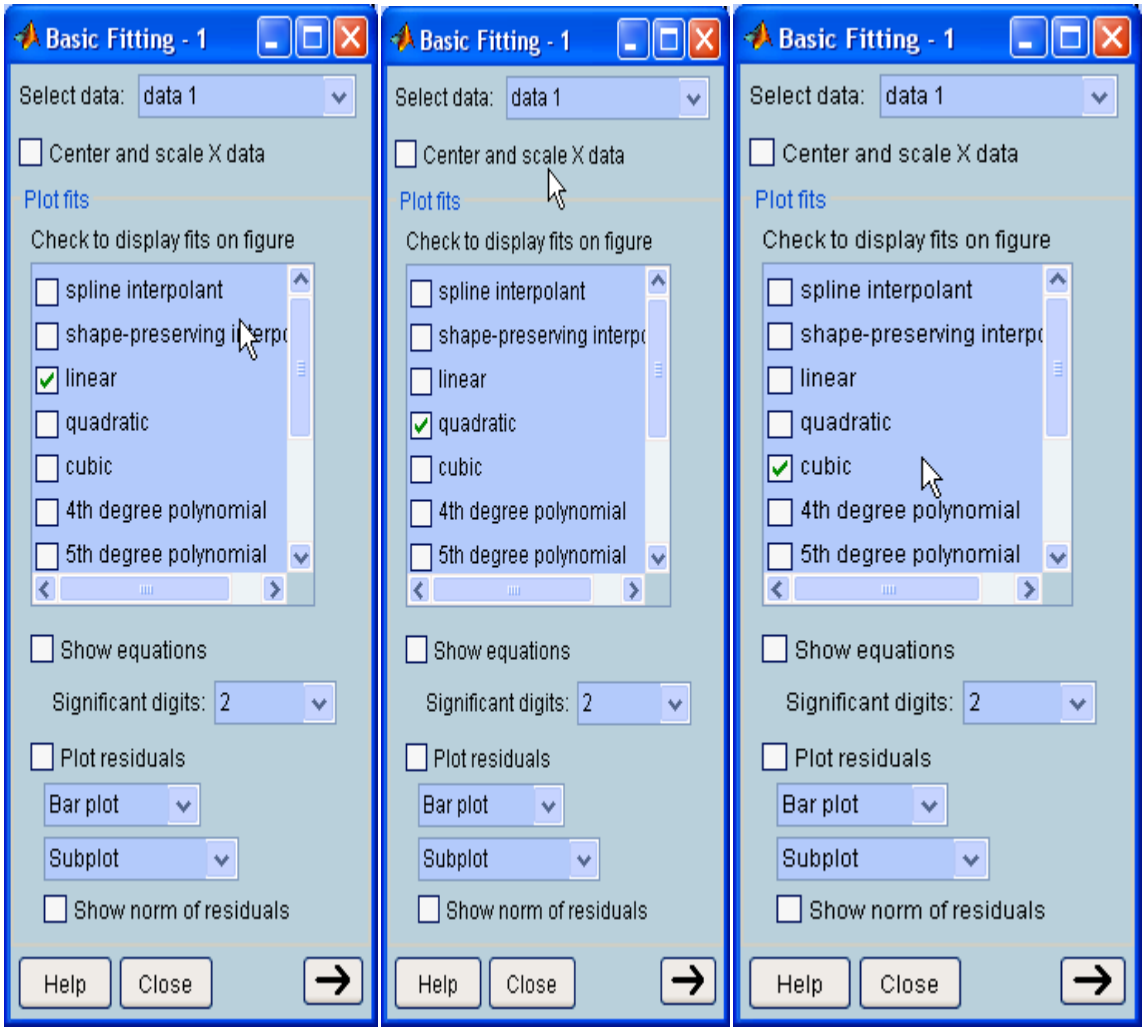
В *MATLAB* 6.0 совмещение функций аппроксимации с графической визуализацией доведено до логического конца — предусмотрена аппроксимация рядом методов точек функции, график которой построен. И все это выполняется прямо в окне редактора графики

Property Editor. Для этого в позиции *Tools* графического окна имеются две новые команды:

Basic Fitting - основные виды аппроксимации (регрессии);

Data Statistics - статистические параметры данных.

Команда *Basic Fitting* открывает окно, дающее доступ к ряду видов аппроксимации и регрессии: сплайновой, эрмитовой и полиномиальной со степенями от 1 (линейная аппроксимация) до 10 (рис. 12, *а*). В том числе со степенью 2 (квадратичная аппроксимация) (рис. 12, *б*) и 3 (кубическая аппроксимация) (рис. 12, *в*).



а) б) в)

Рис. 12. Окно, показывающее различные виды аппроксимации

Команда *Data Statistics* открывает окно с результатами простейшей статистической обработки данных.

3. Оценка погрешности аппроксимации

Средства обработки данных из графического окна позволяют строить столбцовый или линейчатый график погрешностей в узловых точках и наносить на эти графики норму погрешности. Норма дает статистическую оценку среднеквадратической погрешности. Чем она меньше, тем точнее аппроксимация. Для вывода графика погрешности надо установить птичку у параметра *Plot residuals* (график погрешностей) и в меню ниже этого параметра выбрать тип графика (рис. 13).

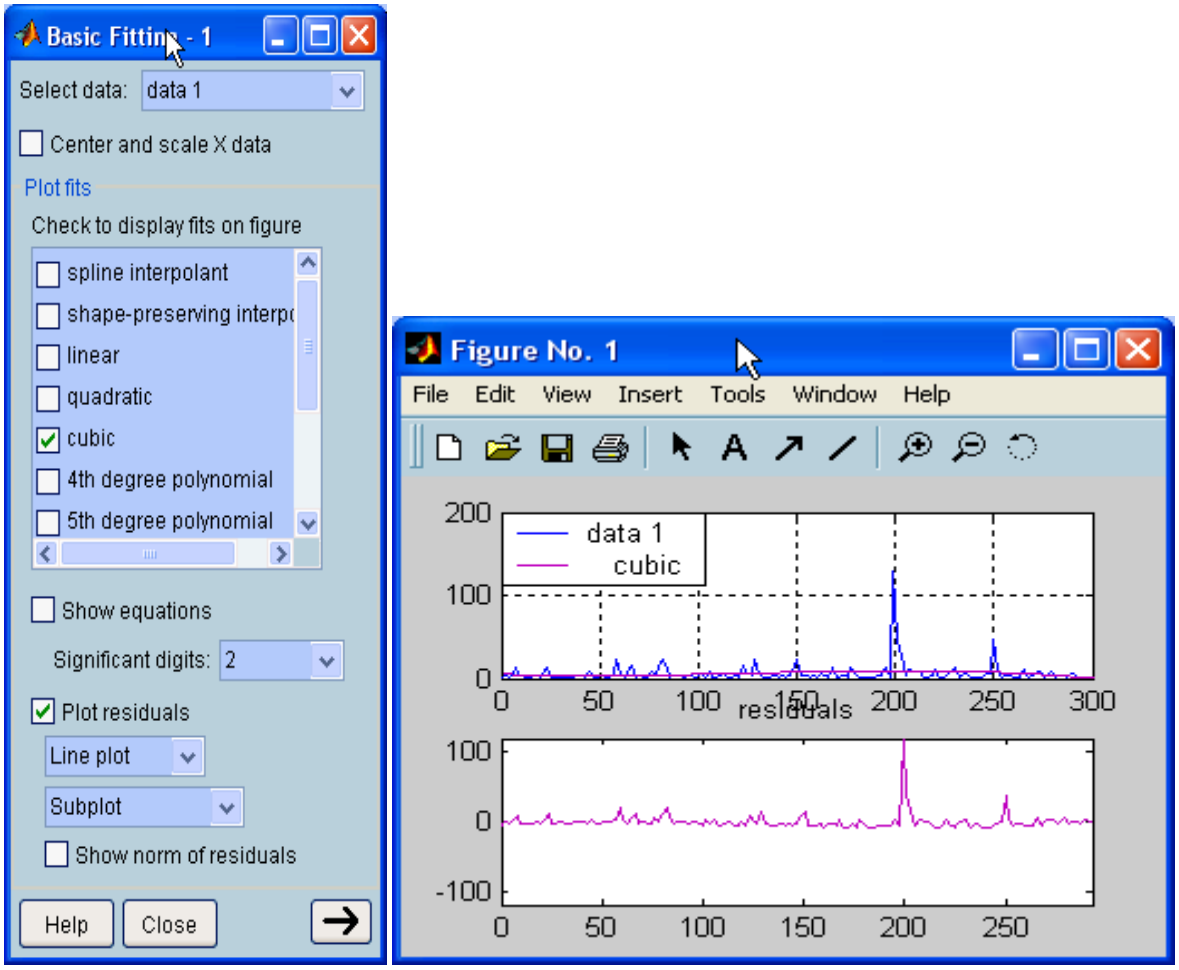


Рис. 13. Окно выбора оценки погрешности аппроксимации

Таким образом, интерфейс графического окна позволяет выполнять эффективную обработку данных наиболее распространенными способами.

Содержание отчета

Отчет должен содержать

1. График формы зашумленного сигнала.
2. График погрешностей измерений.

Контрольные вопросы

1. Опишите функцию одномерного прямого преобразования Фурье.
2. Поясните результаты спектральной плотности сигнала.
3. Оцените погрешность аппроксимации.

Лабораторная работа № 5

ИССЛЕДОВАНИЕ ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ НА ПРИМЕРЕ ПРОГРАММЫ lr2 ДЛЯ MATLAB

Цель работы: изучение дискретного преобразования Фурье для различных видов сигналов, их частоты и амплитуды. Анализ результатов, полученных программой *sigdemo1*.

Теоретическая часть

Для чего нужно быстрое преобразование Фурье или вообще дискретное преобразование Фурье (ДПФ)? Давайте попробуем разобраться.

Пусть у нас есть функция синуса $x = sin(t)$ (рис. 14).

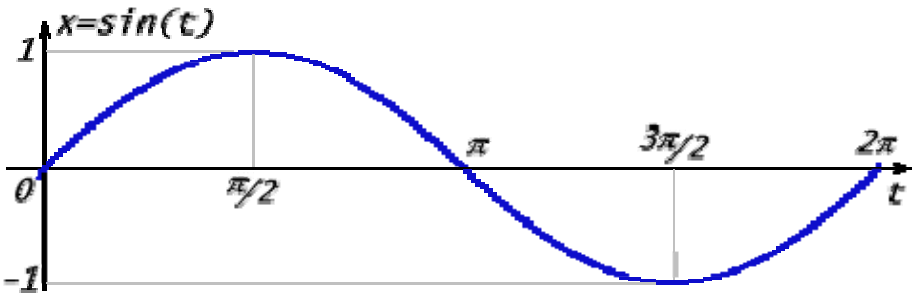


Рис. 14. График функции $x = sin\ (t)$

Максимальная амплитуда этого колебания равна 1. Если умножить его на некоторый коэффициент A, то получим тот же график, растянутый по вертикали в A раз: $x = Asin(t)$. Период колебания равен 2π. Если мы хотим увеличить период до T, то надо умножить переменную t на коэффициент. Это вызовет растяжение графика по горизонтали $x = A\ sin(2\pi t/T)$. Частота колебания обратна периоду $v = 1/T$. Также говорят о круговой частоте, которая вычисляется по формуле $\omega = 2\pi\nu = 2\pi T$. Откуда $x = A\ sin(\omega t)$. И, наконец, есть фаза, обозначаемая как φ. Она определяет сдвиг графика колебания влево. В результате сочетания всех этих параметров получается гармоническое колебание, или просто гармоника (рис. 15).

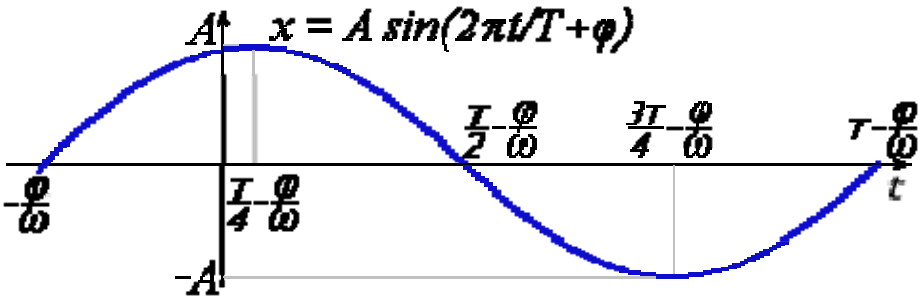


Рис. 15. График функции $x = A\ sin(2\pi t/T+\varphi)$

Очень похоже выглядит и выражение гармоники через косинус (рис. 16).

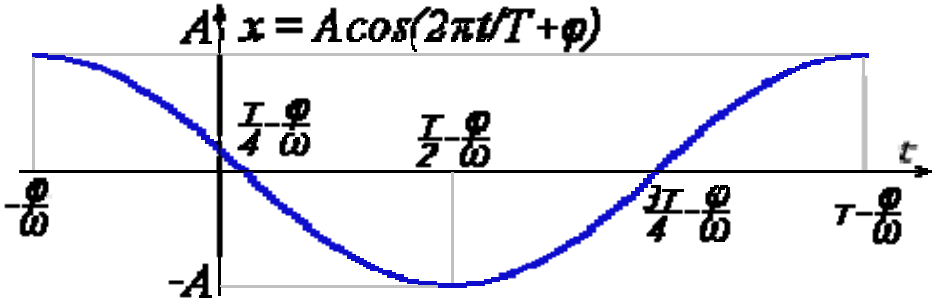


Рис. 16. График функции $x = A\ cos(2\pi t/T+\varphi)$

Большой разницы нет. Достаточно изменить фазу на $\pi/2$, чтобы перейти от синуса к косинусу и обратно. Далее будем подразумевать под гармоникой функцию косинуса

$$x = A\ cos(2\pi t/T + \varphi) = A\ cos(2\pi\nu t + \varphi) = A\ cos(\omega t + \varphi).$$

В природе и технике колебания, описываемые подобной функцией, чрезвычайно распространены, например маятник, струна, водные и звуковые волны и прочее.

Преобразуем (1) по формуле косинуса суммы

$$x = A \cos \varphi \cos(2\pi t/T) - A \sin \varphi \sin(2\pi t/T).$$

Выделим в формуле элементы, независимые от t, и обозначим их как Re и Im:

$$x = Re \cos(2\pi t/T) - Im \sin(2\pi t / T);$$

$$Re = A \cos \varphi, \; Im = A \sin \varphi.$$

По величинам Re и Im можно однозначно восстановить амплитуду и фазу исходной гармоники

$$\varphi = arctg\left(\frac{Im}{Re}\right) \text{ и } A = \sqrt{Re^2 + Im^2}.$$

Теперь возьмем обратное преобразование Фурье

$$x_n = \frac{1}{N}\sum_{k=0}^{N-i} X_k e^{\frac{j2\pi kn}{N}}.$$

Выполним над этой формулой следующие действия: разложим каждое комплексное X_k на мнимую и действительную составляющие $X_k = Re_k + j\, Im_k$; разложим экспоненту по формуле Эйлера на синус и косинус действительного аргумента; перемножим; внесем $1/N$ под знак суммы и перегруппируем элементы в две суммы:

$$\begin{aligned}
x_n &= \frac{1}{N}\sum_{k=0}^{N-1} X_k\left[\cos\left(\frac{2\pi kn}{N}\right) + j\sin\left(\frac{2\pi kn}{N}\right)\right] = \\
&= \frac{1}{N}\sum_{k=0}^{N-1}(Re_k + j\, Im_k)\left[\cos\left(\frac{2\pi kn}{N}\right) + j\sin\left(\frac{2\pi kn}{N}\right)\right] = \\
&= \sum_{k=0}^{N-1}\frac{1}{N}\left[Re_k \cos\left(\frac{2\pi kn}{N}\right) - Im_k \sin\left(\frac{2\pi kn}{N}\right) + \right. \\
&\left. + j\, Im_k \cos\left(\frac{2\pi kn}{N}\right) + j\, Re_k \sin\left(\frac{2\pi kn}{N}\right)\right] = \\
&= \sum_{k=0}^{N-1}\left[\frac{Re_k}{N}\cos\left(\frac{2\pi kn}{N}\right) - \frac{Im_k}{N}\sin\left(\frac{2\pi kn}{N}\right)\right] + \\
&+ j\sum_{k=0}^{N-1}\left[\frac{Re_k}{N}\sin\left(\frac{2\pi kn}{N}\right) + \frac{Im_k}{N}\cos\left(\frac{2\pi kn}{N}\right)\right]
\end{aligned} \qquad (2)$$

Оставим эту формулу пока в стороне и рассмотрим очень распространенную ситуацию. Пусть у нас есть звуковое или какое-то иное колебание в виде функции $x = f(t)$. Пусть это колебание было записано в

виде графика для отрезка времени *[0, T]* (рис. 17). Для обработки компьютером нужно выполнить дискретизацию. Отрезок делится на *N-1* частей и сохраняются значения функции $x_0, x_1, x_2,..., x_N$ для *N* точек на границах отрезков $t_0 = 0$, $t_1 = T/N$, $t_2 = 2T/N,..., t_n = nT/N,..., t_N = T$.

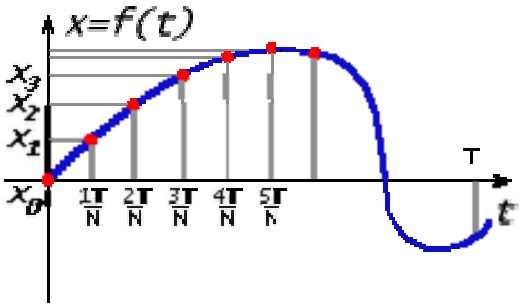


Рис. 17. График функции $x = f(t)$ на отрезке времени *[0, T]*

В результате прямого дискретного преобразования Фурье были получены *N* значений для X_k

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}$$

Теперь, если применить обратное дискретное преобразование Фурье, то получится исходная последовательность *{x}*. Исходная последовательность состояла из действительных чисел, а последовательность *{X}* в общем случае комплексная. Теперь вернемся к формуле (2). Слева стоит действительное число x_n, а справа - две суммы, одна из которых помножена на мнимую единицу *j*. Сами же суммы состоят из действительных слагаемых. Отсюда следует, что вторая сумма равна нулю, если исходная последовательность *{x}* была действительной. Отбросим ее и получим

$$x_n = \sum_{k=0}^{N-1}\left[\frac{Re_k}{N}\cos\left(\frac{2\pi kn}{N}\right) - \frac{Im_k}{N}\sin\left(\frac{2\pi kn}{N}\right)\right]$$

Поскольку при дискретизации мы брали $t_n = nT/N$ и $x_n = f(t_n)$, то можем выполнить замену: $n = t_n N/T$. Следовательно, в синусе и косинусе вместо $2\pi kn/N$ можно написать $2\pi kt_n/T$. В результате получим

$$x_n = f(t_n) = \sum_{k=0}^{N-1}\left[\frac{Re_k}{N}\cos\left(\frac{2\pi kt_n}{T}\right) - \frac{Im_k}{N}\sin\left(\frac{2\pi kt_n}{T}\right)\right] \qquad (3)$$

Сопоставим эту формулу с формулами (4) и (5) для гармоники

$$x = A\cos(2\pi t/T + \varphi) = A\cos(2\pi\nu t + \varphi) = A\cos(\omega t + \varphi); \qquad (4)$$

$$x = Re\cos(2\pi t/T) - Im\sin(2\pi t / T). \qquad (5)$$

Мы видим, что сумма (3) представляет собой сумму из *N* гармонических колебаний разной частоты, фазы и амплитуды:

$$f(t_n) = \sum_{k=0}^{N-1} A_k\cos(2\pi t_n/T_k + \varphi_k) = \sum_{k=0}^{N-1} A_k\cos(2\pi t_n\nu_k + \varphi_k) = \sum_{k=0}^{N-1} G_k(t_n)$$

Далее будем функцию

$$G_k(t) = A_k \cos(2\pi tk/T + \varphi_k)$$

называть *k*-й гармоникой.

Амплитуда, фаза, частота и период каждой из гармоник связаны с коэффициентами X_k формулами

$$X_k = Re_k + j\,Im_k$$

$$X_k = NA_k e^{i\varphi_k}$$

$$A_k = \frac{1}{N}\sqrt{Re_k^2 + Im_k^2}$$

$$\varphi_k = arctg\left(\frac{Im_k}{Re_k}\right)$$

$$\nu_k = \nu k \qquad T_k = T/k$$

Физический смысл дискретного преобразования Фурье состоит в том, чтобы представить некоторый дискретный сигнал в виде суммы гармоник. Параметры каждой гармоники вычисляются прямым преобразованием, а сумма гармоник - обратным. Рабочее окно программы *lr2* представлено на рис. 18.

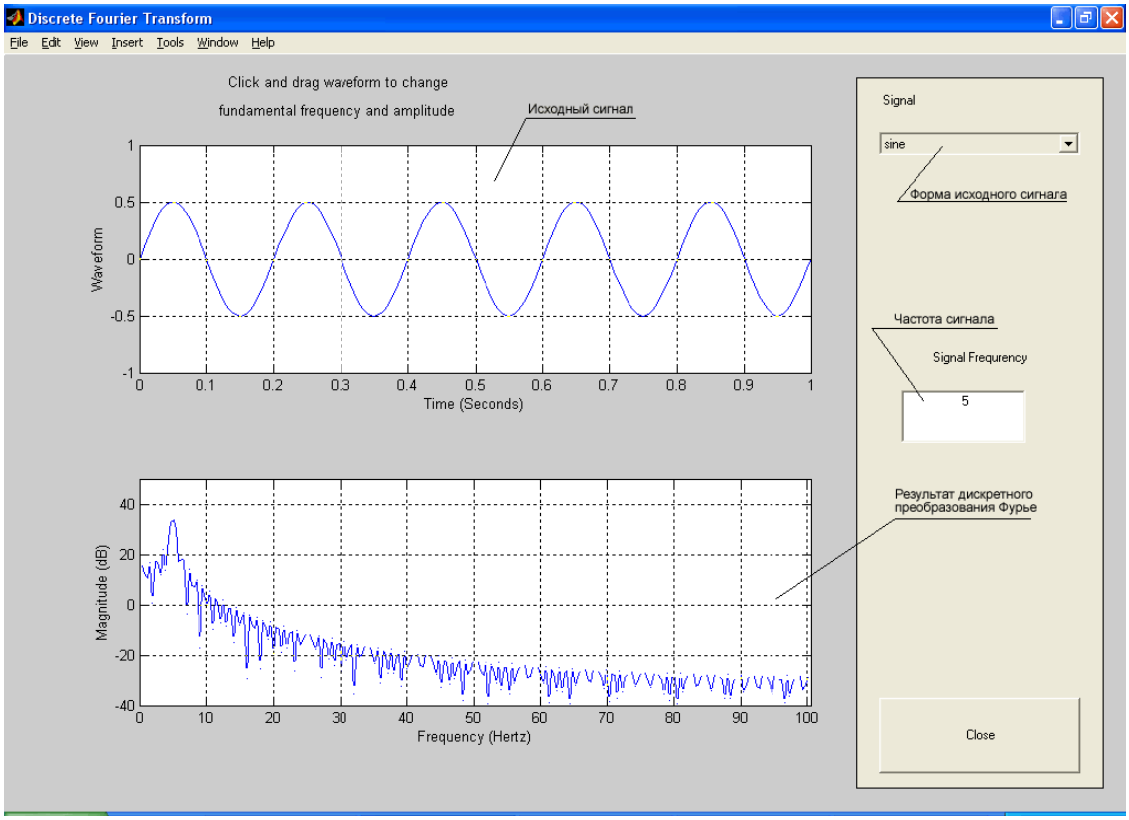


Рис. 18. График дискретного преобразования Фурье

Порядок выполнения работы

1. Запустите программу *lr2* из command window программы *matlab.*

2. Выберите *sine* форму сигнала. Установите параметр частоты сигнала в 5 Гц, амплитуда – 0,5 отн. ед. Зарисуйте результат дискретного преобразования Фурье.

3. Для амплитуды 0,5 отн. ед. получить результат дискретного преобразования Фурье (ДПФ) для частот 5, 10, 25, 50 и 90 Гц.

4. Изменить значение амплитуды на 1 и построить графики для 5, 10, 25, 50 и 90 Гц.

5. Произвести ДПФ с аналогичными значениями для квадратной и пилообразной форм сигнала.

Содержание отчета

Отчет должен содержать

1. Графики дискретного преобразования Фурье для сигнала *sine* формы при различных значениях частот сигнала.

2. Графики дискретного преобразования Фурье для сигнала *sine* формы при различных значениях амплитуд сигнала.

3. ДПФ с аналогичными значениями для квадратной и пилообразной форм сигнала.

Контрольные вопросы

1. Чему равна максимальная амплитуда колебания?

2. В чем состоит физический смысл дискретного преобразования Фурье?

Лабораторная работа № 6

ИССЛЕДОВАНИЕ ПРИНЦИПОВ МОДУЛЯЦИИ СИГНАЛОВ НА ПРИМЕРЕ ПРОГРАММЫ lr1 ДЛЯ MATLAB

Цель работы: изучение модуляции различных сигналов с практическим исследованием их характеристик.

Теоретическая часть

Сигналы от измерительных датчиков и любых других источников сигналов передаются по линиям связи к измерительным приборам и в измерительно-вычислительные центры регистрации и обработки данных. Как правило, информационные сигналы являются низкочастот-

ными и ограниченными по ширине спектра в отличие от широкополосных высокочастотных каналов связи, рассчитанных на передачу сигналов от множества источников одновременно с частотным разделением каналов. Перенос спектра сигналов из низкочастотной области в выделенную для их передачи область высоких частот канала связи выполняется операцией модуляции.

Допустим, что низкочастотный сигнал, подлежащий передаче по какому-либо каналу связи или радиоканалу, задается функцией *s(t)*. В канале связи выделяется для передачи данного сигнала определенный высокочастотный диапазон. На входе канала связи в специальном передающем устройстве формируется вспомогательный, как правило, непрерывный во времени периодический высокочастотный сигнал $u(t) = f(t; a_1, a_2, \ldots a_m)$. Совокупность параметров a_i определяет форму вспомогательного сигнала. Если на один из этих параметров перенести сигнал *s(t*), т.е. сделать его значение пропорционально зависимым от значения *s(t)* во времени (или по любой другой независимой переменной), то форма сигнала *u(t*) приобретает новое свойство. Она несет информацию, тождественную информации в сигнале *s(t).* Именно поэтому такой сигнал называют несущим сигналом, несущим колебанием или просто несущей *(carrier),* а физический процесс переноса информации на параметры несущего сигнала – его модуляцией *(modulation).* Исходный информационный сигнал называют модулирующим *(modulating signal),* результат модуляции – модулированным сигналом *(modulated signal).* Обратную операцию выделения модулирующего сигнала из модулированного колебания называют демодуляцией *(demodulation).*

В качестве несущих сигналов обычно используются гармонические колебания $u(t) = U \cdot \cos(\omega t + \varphi)$, которые имеют три свободных параметра: U, ω и φ. В зависимости от того, на какой из данных параметров переносится информация, различают амплитудную, частотную или фазовую модуляцию несущего сигнала. Частотная и фазовая модуляция тесно взаимосвязаны, поскольку изменяют аргумент функции косинуса, и часто их объединяют под общим названием угловая модуляция *(angle modulation).* В каналах передачи цифровой информации получила также распространение квадратурная модуляция, при которой одновременно изменяются амплитуда и фаза несущих колебаний.

Амплитудная модуляция *(amplitude modulation, AM)* исторически была первым освоенным на практике видом модуляции. В настоящее время *AM* применяется в основном только для радиовещания на сравнительно низких частотах (не выше коротких волн) и для передачи изображения в телевизионном вещании. Это обусловлено низким КПД использования энергии модулированных сигналов.

AM соответствует переносу информации $s(t) \Rightarrow U(t)$ при постоянных значениях параметров ω и φ. *AM*-сигнал представляет собой произведение информационной огибающей *U(t)* и гармонического колебания ее заполнения с более высокими частотами. Форма записи амплитудно-модулированного сигнала

$$u(t) = U(t)\cos(\omega_o t + \varphi_o),$$
$$U(t) = U_m[1 + M \cdot s(t)],$$

где U_m – постоянная амплитуда несущего колебания при отсутствии входного (модулирующего) сигнала *s(t)*, *M* – коэффициент амплитудной модуляции.

Значение *M* характеризует глубину амплитудной модуляции. В простейшем случае, если модулирующий сигнал представлен одночастотным гармоническим колебанием с амплитудой S_o, то коэффициент модуляции равен отношению амплитуд модулирующего и несущего колебания $M = S_o/U_m$. Значение *M* должно находиться в пределах от 0 до 1 для всех гармоник модулирующего сигнала. При значении $M < 1$ форма огибающей несущего колебания полностью повторяет форму модулирующего сигнала *s(t),* что можно видеть на рис. 19 (сигнал $s(t) = \sin(\omega_s t)$). Малую глубину модуляции для основных гармоник модулирующего сигнала ($M << 1$) применять нецелесообразно, так как при этом мощность передаваемого информационного сигнала будет много меньше мощности несущего колебания, и мощность передатчика используется неэкономично.

На рис. 20 приведен пример так называемой глубокой модуляции, при которой значение *M* стремится к 1 в экстремальных точках функции *s(t)*. При глубокой модуляции используются также понятия относительного коэффициента модуляции вверх: $M_в = (U_{max} - U_m)/U_m$ и модуляции вниз: $M_н = (U_m - U_{min})/U_m$, которые обычно выражаются в процентах.

Стопроцентная модуляция (M=1) может приводить к искажениям сигналов при перегрузках передатчика, если последний имеет ограниченный динамический диапазон по амплитуде несущих частот или ограниченную мощность передатчика (увеличение амплитуды несущих колебаний в пиковых интервалах сигнала *U(t)* в два раза требует увеличения мощности передатчика в четыре раза).

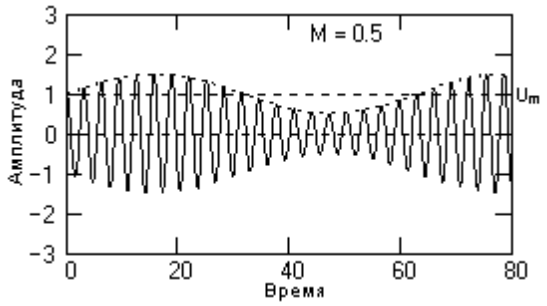


Рис. 19. Модулированный сигнал

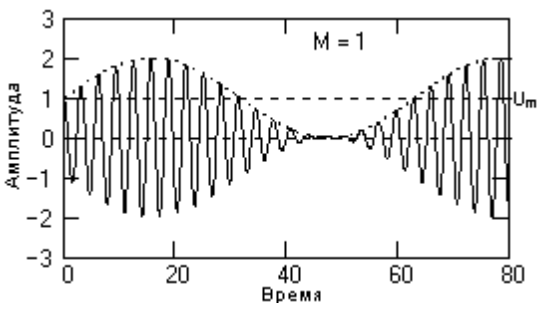


Рис. 20. Глубокая модуляция

При M>1 возникает так называемая перемодуляция, пример которой приведен на рис. 21. Форма огибающей при перемодуляции искажается относительно формы модулирующего сигнала и после демодуляции, если применяются ее простейшие методы, информация может искажаться.

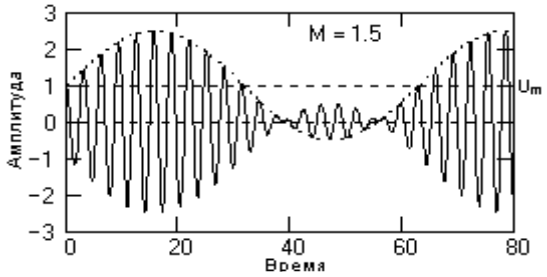


Рис. 21. Демодуляция сигнала

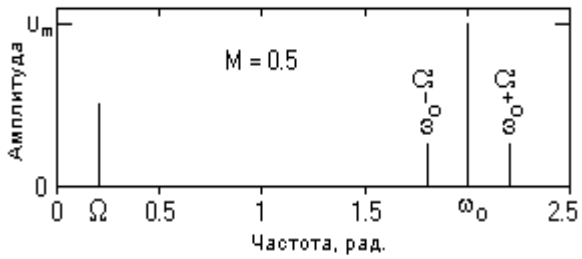


Рис. 22. Физические спектры сигналов

Фазовая модуляция (ФМ, *phase modulation - PM*). При фазовой модуляции значение фазового угла постоянной несущей частоты колебаний ω_o пропорционально амплитуде модулирующего сигнала *s(t)* (рис. 22). Соответственно уравнение ФМ-сигнала определяется выражением

$$u(t) = U_m \cos[\omega_o t + k \cdot s(t)],$$

где k – коэффициент пропорциональности. Пример однотонального ФМ-сигнала приведен на рис. 23.

При $s(t) = 0$, ФМ-сигнал является простым гармоническим колебанием и показан на рисунке функцией $u_o(t)$. С увеличением значений $s(t)$ полная фаза колебаний $\psi(t)=\omega_o t+k\cdot s(t)$ нарастает во времени быстрее и опережает линейное нарастание $\omega_o t$. Соответственно при уменьшении значений $s(t)$ скорость роста полной фазы во времени спадает. В моменты экстремальных значений $s(t)$ абсолютное значение фазового сдвига $\Delta\psi$ между ФМ-сигналом и значением $\omega_o t$ немодулированного колебания также является максимальным и носит название девиации фазы (вверх $\Delta\varphi_в = k\cdot s_{max}(t)$ или вниз $\Delta\varphi_н = k\cdot s_{min}(t)$ с учетом знака экстремальных значений модулирующего сигнала).

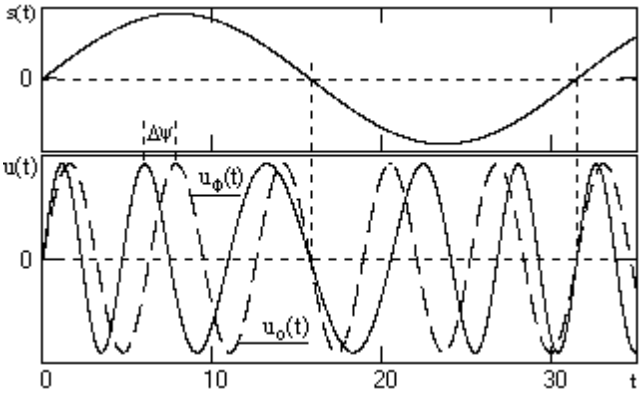


Рис. 23. Фазомодулированный сигнал

Для колебаний с угловой модуляцией применяется также понятие мгновенной частоты (instantaneous frequency), под которой понимают производную от полной фазы по времени

$$\omega(t) = \psi(t)/dt = \omega_o + k\, ds(t)/dt.$$

Полная фаза колебаний в произвольный момент времени может быть определена интегрированием мгновенной частоты

$$\psi(t) = \int_{-\infty}^{t} \omega(t)\, dt, \quad или \quad \psi(t) = \int_{0}^{t} \omega(t)\, dt + \varphi_o.$$

Частотная модуляция (ЧМ, *frequency modulation - FM*) характеризуется линейной связью модулирующего сигнала с мгновенной частотой колебаний, при которой мгновенная частота колебаний образуется сложением частоты высокочастотного несущего колебания ω_o со значением амплитуды модулирующего сигнала с определенным коэффициентом пропорциональности

$$\omega(t) = \omega_o + k\cdot s(t).$$

Соответственно полная фаза колебаний

$$\psi(t) = \omega_o(t) + k\int_{-\infty}^{t} s(t)\,dt, \quad или \quad \psi(t) = \omega_o(t) + k\int_{0}^{t} s(t)\,dt + \varphi_o.$$

Уравнение ЧМ-сигнала

$$u(t) = U_m \cos(\omega_o t + k\int_{0}^{t} s(t)\,dt + \varphi_o).$$

Аналогично ФМ для характеристики глубины частотной модуляции используются понятия девиации частоты вверх $\Delta\omega_в = k \cdot s_{max}(t)$ и вниз $\Delta\omega_н = k \cdot s_{min}(t)$.

Частотная и фазовая модуляции взаимосвязаны. Если изменяется начальная фаза колебания, изменяется и мгновенная частота и наоборот. По этой причине их и объединяют под общим названием угловой модуляции (УМ). По форме колебаний с угловой модуляцией невозможно определить, к какому виду модуляции относится данное колебание: к ФМ или ЧМ, а при достаточно гладких функциях $s(t)$ формы сигналов ФМ и ЧМ вообще практически не отличаются.

На рис. 24 изображено рабочее окно программы *lr1*. Выносными линиями обозначены основные элементы интерфейса.

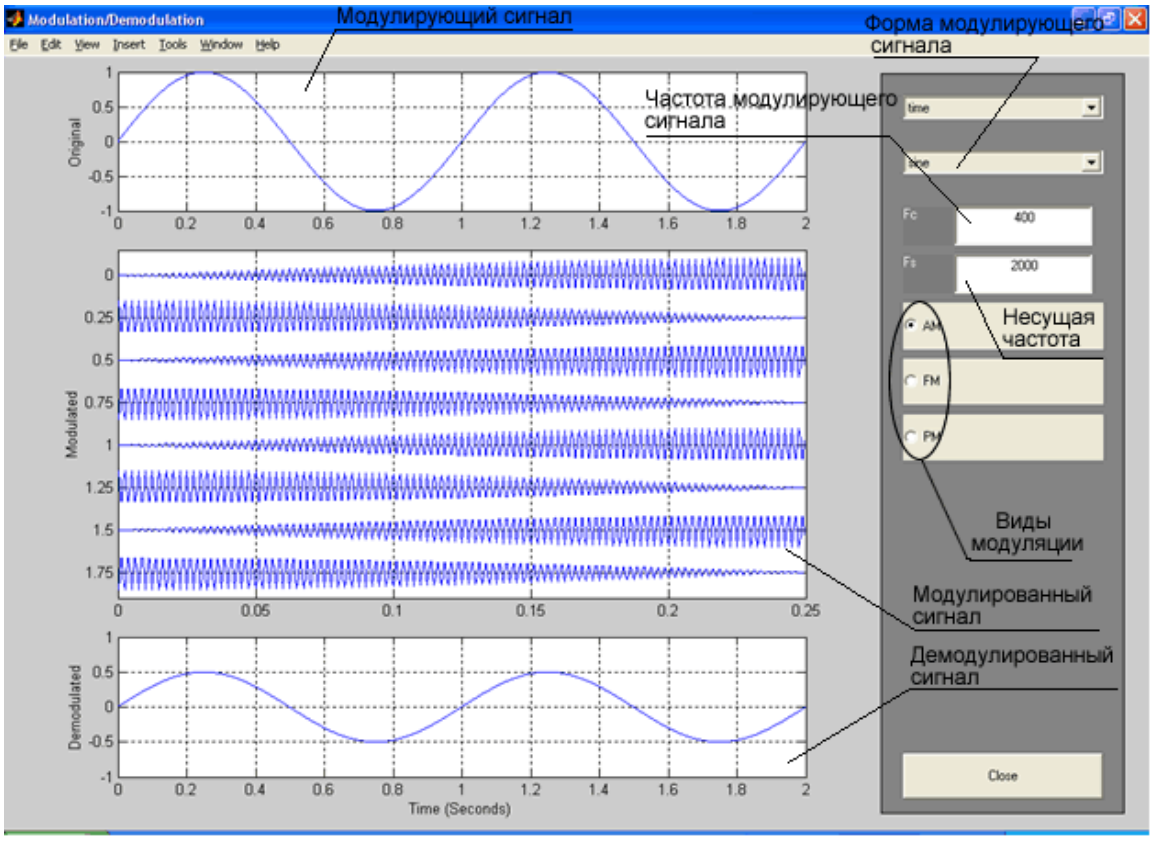


Рис. 24. Рабочее окно программы lr1

Форма исходного сигнала выбирается из соответствующего списка, состоящего из подпунктов: *speech* (речь), *sine* (синусоидальная форма сигнала), *square* (квадратная форма сигнала) и *triangle* (треугольная форма). В поле *Fc* задается частота передаваемого сигнала, а в поле *Fc* – несущая частота.

Выбор методов модуляции осуществляется точечным переключателем. *AM* соответствует амплитудной, *FM* – частотной, а *PM* – фазовой модуляции. Стоит обратить внимание на график модулированного сигнала. Он представляет собой масштабное представление сигнала с разбиением на несколько строк.

Порядок выполнения работы

Ознакомиться с основными принципами модуляции сигналов и интерфейсом программы *lr1*.

1. Запустить программу *Matlab* с помощью ярлыка на рабочем столе или из меню Пуск программы-*Matlab Realese N-matlab.* Дождавшись загрузки приложения в окне *"command window",* набрать lr1 и нажать enter.
2. В поле «форма модулирующего сигнала» выбрать sine.
3. Установить значение модулирующей частоты 300 Гц, а несущей 1500 Гц.
4. Зарисовать полученные графики модулированного сигнала для *AM, FM* и *PM* модуляции.
5. Построить графики для частот *Fc* = 600, Гц, *Fs* = 3000 Гц и *Fc* = 1500 Гц, *Fs* = 17000 Гц.
6. Построить графики для квадратной и треугольной форм сигнала со значениями *Fc* и *Fs*, указанными для *sine*.
7. Сделать выводы по графикам.

Содержание отчета

Отчет должен содержать

1. Графики модулированного сигнала для *AM, FM* и *PM* модуляции.
2. Графики для частот *Fc* = 600 Гц, *Fs* = 3000 Гц, *Fc* = 1500 Гц, *Fs* = 17000 Гц.

Контрольные вопросы

1. Раскройте понятие «модуляция».
2. Как выглядят графики квадратной и треугольной форм сигнала со значениями *Fc* и *Fs*, указанными для *sine*?

Лабораторная работа № 7

ПРОЕКТИРОВАНИЕ ЦИФРОВОГО ФИЛЬТРА С КВАНТОВАНИЕМ ПАРАМЕТРОВ

Цель работы: провести расчет и квантование в среде *Matlab* для эллиптического полосового фильтра с заданными параметрами.

Теоретическая часть

Основные принципы реализации цифровых фильтров рассмотрены в лабораторной работе «Проектирование цифрового фильтра в среде matlab».

Для иллюстрации и обсуждения эффектов квантования решим следующую задачу: спроектировать цифровой эллиптический полосовой фильтр, требования к которому заданы в табл. 2. Для этого нам необходимо выделить и заполнить соответствующие позиции на странице *Filter Design* главного окна *fdatool*, как показано на рис. 25, после чего нажать на клавишу *Design Filter*, расположенную внизу окна. После завершения расчёта в верхней части окна будут представлены результаты (рис. 26). Как видно из рисунка, в области отображения характеристик и параметров выводится АЧХ полученного фильтра, однако пользователь может по мере необходимости для вывода переключать её содержимое, например, импульсной характеристики, полюсов и нулей и так далее.

До настоящего момента все операции над данными выполнялись с машинной точностью, когда для представления чисел использовался формат *double* (в соответствии с этим форматом для записи и хранения числа отводится 8 байт). Теперь проанализируем, что произойдёт при изменении формата представления чисел. Такая потребность возникает, когда необходимо выполнить эмуляцию работы построенного фильтра на базе целевого процессора, использующего отличные от *PC* форматы данных. В главном окне, ниже раздела *Current Filter Information*, находится раздел *Quantization*, содержащий единственное поле *Turn Quantization On*. Для запуска процедуры расчёта установим

в этом поле флажок, запустив тем самым процедуру расчёта квантованного фильтра с параметрами квантования, заданными по умолчанию. (Также по умолчанию расчёт выполняется для прямой реализации фильтра).

По окончании процедуры расчёта АЧХ нового (квантованного) фильтра наложится на АЧХ фильтра-прототипа, рассчитанного ранее. Область отображения характеристик будет выглядеть так, как показано на рис. 27, где АЧХ фильтра-прототипа обозначена как *Reference*, а квантованного - как *Quantized*. Из графиков видно, что АЧХ фильтра с учётом эффектов квантования (*Quantized*) существенно отличается от соответствующей характеристики фильтра-прототипа (*Reference*). Сохраним полученные результаты на диске, для чего откроем меню *File* и выберем раздел *Save Session As*…

Таблица 2

Параметры для расчета	Значение параметра
Частота дискретизации	48 кГц
Полоса задерживания 1	от 0 до 7,2 кГц
Полоса пропускания	от 8 до 12 кГц
Полоса задерживания 2	от 12,8 до 24 кГц
Минимально допустимое ослабление в полосе задерживания 1	80 дБ
Максимально допустимое ослабление в полосе пропускания	1 дБ
Минимально допустимое подавление в полосе задерживания 2	80 дБ
Порядок фильтра	минимальный для заданных требований

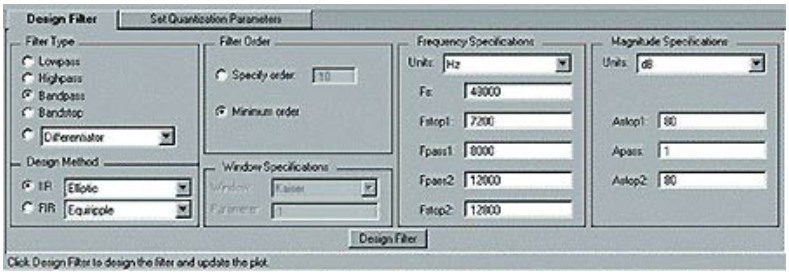


Рис. 25. Страница *Design Filter*, содержащая данные табл. 2

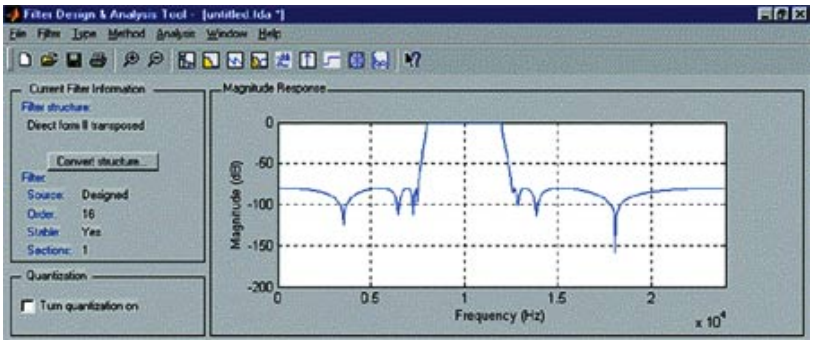


Рис. 26. Амплитудно-частотная характеристика фильтра-прототипа

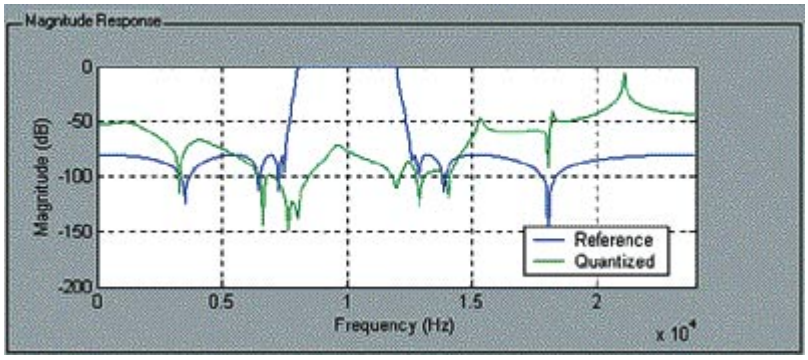


Рис. 27. Амплитудно-частотные характеристики
фильтра-прототипа и квантованного фильтра

Чтобы разобраться в причинах, приведших к таким сильным изменениям, перейдём на страницу *Set Quantization Parameters*, показанную на рис. 28, и проанализируем её содержимое. Как следует из рисунка, имеется пять видов объектов, которые подвергаются квантованию

- *Coefficient* (коэффициенты фильтра);
- *Input* (входные сигналы);
- *Output* (выходные сигналы);
- *Multiplicand* (множимые);
- *Product* (произведения);
- *Sum* (суммы).

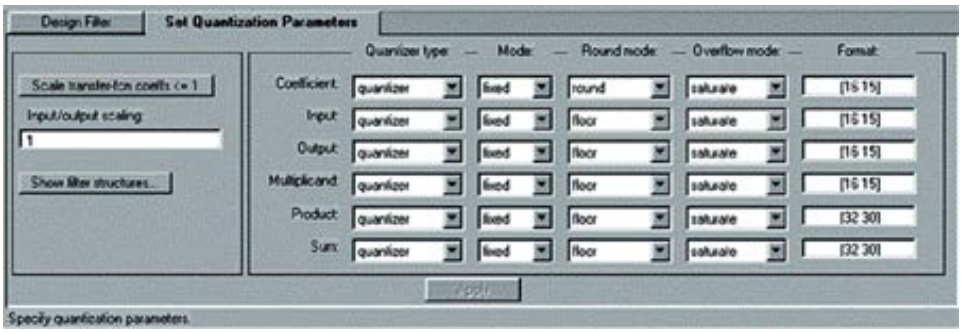


Рис. 28. Общий вид страницы *Set Quantization Parameters*

Для выполнения операции квантования используются квантователи перечисленных объектов. Таким образом, квантованный фильтр представляет собой нелинейную цифровую систему, включающую:

- квантователь входного сигнала;
- собственно фильтр с квантованными значениями коэффициентов, квантователями множимых (сигналов на входах умножителей), произведений (сигналов на выходах умножителей) и сумм (сигналов на выходах сумматоров);
- квантователь выходного сигнала.

Пример квантованного фильтра 2-го порядка показан на рис. 29, где введены следующие обозначения:

- *q1* - квантователь коэффициентов;
- *q2* - квантователь входного сигнала;
- *q3* - квантователь выходного сигнала;
- *q4* - квантователь множимого;
- *q5* - квантователь произведения;
- *q6* - квантователь суммы.

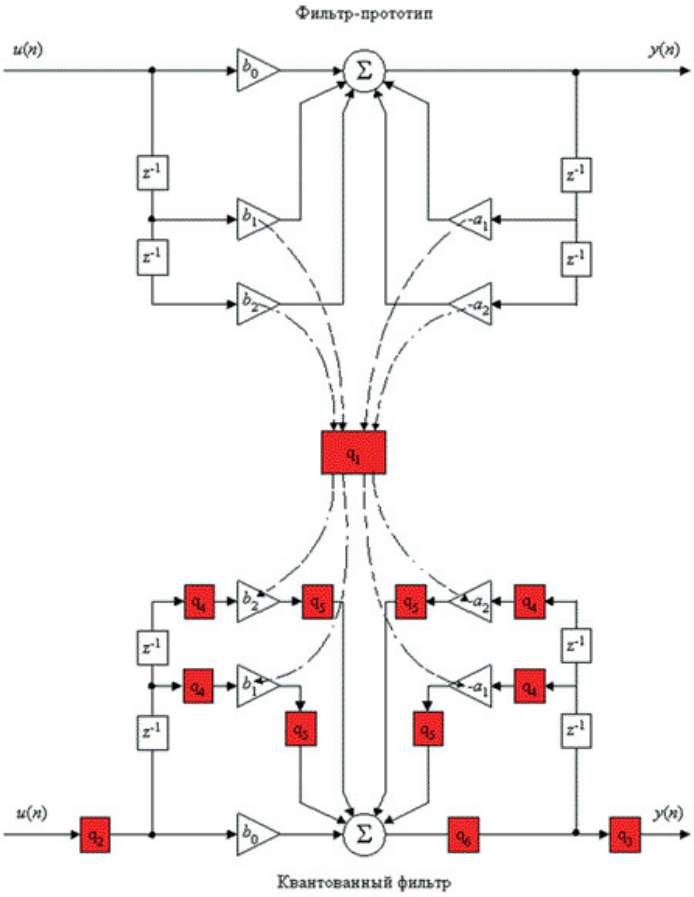


Рис. 29. Структурные схемы фильтра-прототипа
и квантованного фильтра 2-го порядка

Свойства квантованного фильтра в целом зависят от параметров каждого из перечисленных квантователей. Эти параметры определяются содержимым пяти колонок, или полей, расположенных правее имён объектов квантования. Параметры определяют, каким образом формируется сигнал или вычисляется коэффициент на выходе соответствующего квантователя.

Говоря о квантователях, прежде всего подчеркнём тот очевидный факт, что реализация любого цифрового фильтра основывается на использовании арифметики с фиксированной или плавающей точкой (*Fixed-Point* или *Floating-Point Arithmetic* соответственно). В связи с этим остановимся кратко на форматах представления чисел и особенностях выполнения арифметических операций в программе *fdatool*.

Арифметика с фиксированной точкой: форматы данных и реализация операций.

Выбор работы каждого из квантователей в режиме с фиксированной или плавающей точкой задаётся значениями параметров колонки *Mode,* расположенной на странице *Set Quantization Parameters*. Для реализации арифметики с фиксированной точкой необходимо выбрать значение *fixed.*

Двоичные числа с фиксированной точкой определяются в битах длиной слова *w* и длиной дробной части числа *f*. При этом длина дробной части может быть задана в диапазоне от 0 до *w* - 1 бит. Общее представление числа в формате с фиксированной точкой показано на рис. 30. Пользователь может задавать длину слова до 64 бит включительно, однако побитовое соответствие результатов моделирования с помощью *fdatool* и реальной работы целевого компьютера обеспечивается, если длина слова определена в пределах 53 бит. Если же выбранная длина слова находится в диапазоне 54 $\leq w \leq 64$, то происходит потеря значимости, то есть в младшие биты записываются нули. В пакете *MATLAB* и, в частности, в функциях библиотеки *Filter Design* формат для чисел с фиксированной точкой задаётся в виде *[w, f]* в колонке *format.*

Числа с фиксированной точкой могут быть беззнаковыми или со знаком. В первом случае старший бит, как и остальные, используется

для представления величины числа, тогда как во втором - для его знака (0 соответствует знаку "плюс", 1 - знаку "минус"). Динамический диапазон для беззнаковых чисел равен $[0, 2^{w-f}-2^{-f}]$, а для чисел со знаком - $[-2^{w-f-1}, 2^{w-f-1}-2^{-f}]$. В обоих случаях точность, то есть разность двух ближайших чисел в данном формате, равна $\varepsilon = 2^{-f}$.

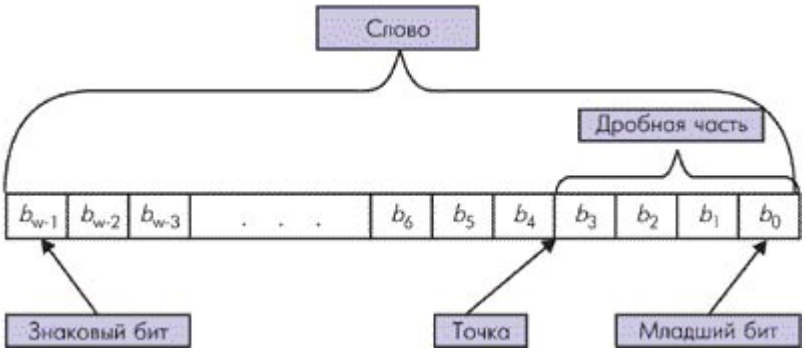


Рис. 30. Общее представление числа в формате с фиксированной точкой

В связи с тем что результатом квантования является замена квантуемой величины числом из конечного, предопределённого набора чисел, называемых уровнями квантования, необходимо ввести информацию о том, каким образом эту замену выполнять (значения и число уровней квантования определяются используемым форматом). Для этого в программе *fdatool* имеется поле *Round mode*, где для каждого квантователя можно выбрать способ округления. Пользователь имеет возможность выбрать один из следующих режимов:

- *ceil* - результат равен значению ближайшего уровня квантования в сторону плюс бесконечности;
- *fix* - результат равен значению ближайшего уровня квантования в сторону нуля;
- *floor* - результат равен значению ближайшего уровня квантования в сторону минус бесконечности;
- *round* - результат равен значению ближайшего уровня квантования; если квантуемая отрицательная величина лежит ровно посередине между уровнями квантования, результатом является значение ближайшего уровня в сторону минус бесконечности;

если квантуемая положительная величина лежит ровно посередине между уровнями квантования, результатом будет значение ближайшего уровня квантования в сторону плюс бесконечности;

- *convergent* - способ округления такой же, как и *round*, однако, если квантуемая величина лежит ровно посередине между уровнями квантования, то округление по правилам *round* выполняется лишь в том случае, когда в младший бит после округления записывается единица.

Когда квантованию подвергается величина, значение которой находится за пределами динамического диапазона квантователя, возникает явление переполнения. Способ обработки таких величин задаётся в поле *Overflow Mode* (режим переполнения) для каждого квантователя. Пользователь может выбрать режим *saturate* или *wrap.*

Режим *saturate* означает разрешение работы с насыщением: квантуемая величина, лежащая вне динамического диапазона, полагается равной ближайшему предельно допустимому числу. В случае выбора режима *wrap* старшие биты квантуемой величины, лежащей вне динамического диапазона, будут отброшены, и результат квантования будет представлен младшими *w*-битами, при этом для чисел со знаком старший бит $b_{>w-1}$ (см. рис. 30).

Рассмотрим следующий пример. Пусть имеются два квантователя - *q1* и *q2*, для которых задан формат *fixed* [3 2], и пусть первый из них работает в режиме *saturate*, а второй — в режиме *wrap*. В этом случае шаг квантования для обоих квантователей ε= 0,25, динамический диапазон - [-1, 0,75]. Если входная квантуемая величина равна 1,25, то на выходах этих квантователей будем иметь:

0,75 для квантователя *q1*; -0,75 для квантователя *q2*.

Аналогично, если квантуемая величина равна -1,25, результатами квантования будут следующие значения:

-1 для квантователя *q1*; 0,75 для квантователя *q2*.

На рис. 31, *а-г* показано, как формируются значения на выходе каждого из квантователей для рассмотренных четырёх случаев.

Функции библиотеки *Filter Designer* позволяют использовать два типа квантователей: *quantizer* и *unitquantizer*. Они работают практиче-

ски одинаково за исключением того, что на выходе квантователя *unitquantizer* будет единица, если на его вход поступает величина из диапазона [1−ε, 1+ε]. Например, пусть тип первого из рассмотренных нами выше квантователей *(q1) quantizer*, а второго *(q2) - unitquantizer*. Если на вход *q1* подать единицу, то возникнет переполнение, так как используется формат *fixed* [3, 2], и на выходе будет значение 0,75 (максимальная величина динамического диапазона квантователя *q1*). Если же единицу подать на вход квантователя *q2*, переполнения не возникнет, а на его выходе будет единица. В *fdatool* выбор типа квантователя осуществляется на странице *Set Quantization Parameters* в колонке *Quantizer Type*.

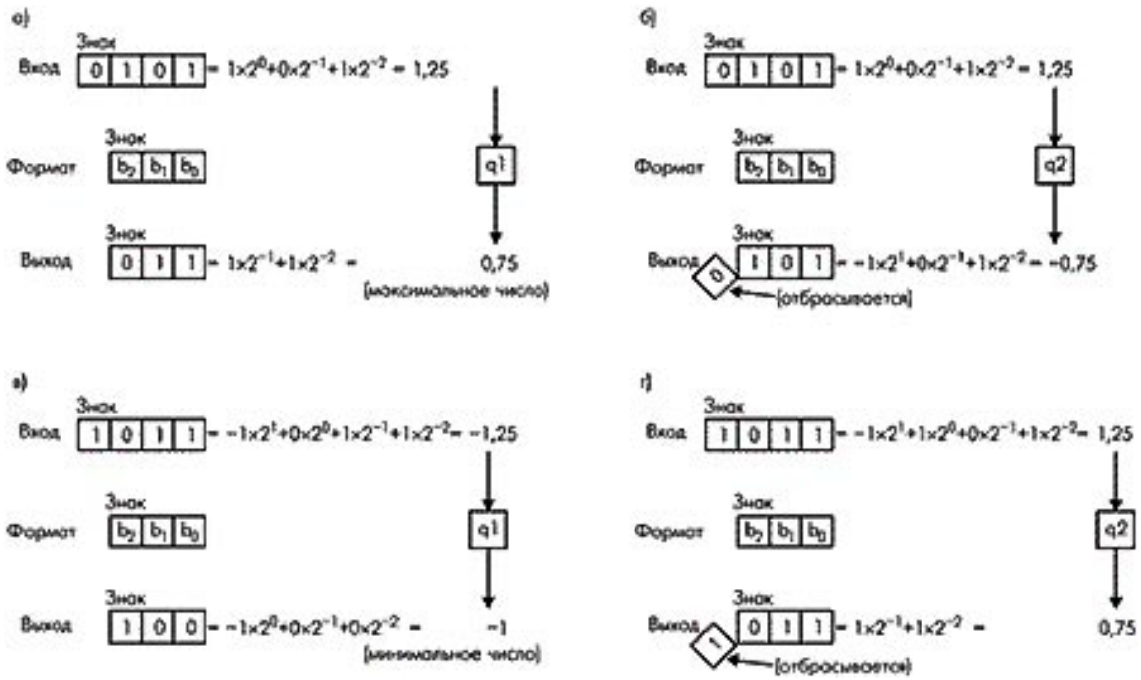


Рис. 31. Варианты квантования в режиме переполнения: положительного числа квантователем *q1* (а), положительного числа квантователем *q2* (б), отрицательного числа квантователем *q1* (в) и отрицательного числа квантователем *q2* (г)

Арифметика с плавающей точкой: форматы данных и реализация операций

Недостатком описания чисел в формате с фиксированной точкой является неудобство представления очень больших и очень маленьких чисел при использовании разумной длины слова *w*. Это ограниче-

ние снимается, если используется формат описания с плавающей точкой. Любое двоичное число с плавающей точкой можно представить в виде $\pm F \cdot 2E$, где F обозначает мантиссу, или дробную часть, 2 - основание системы исчисления, а E - порядок. Длина слова при представлении числа в формате с плавающей точкой - $w = f+e+1$, где f - количество бит, отводимых для хранения мантиссы (длина мантиссы), а e - количество бит, отводимых для хранения порядка (длина порядка). Ещё один бит используется для записи знака числа, он обозначается символом *s (sign)*. Знаковый бит положительного числа содержит нуль *(s = 0)*, отрицательного - единицу *(s = 1)*. Общее представление числа в формате с плавающей точкой показано на рис 32, *а*.

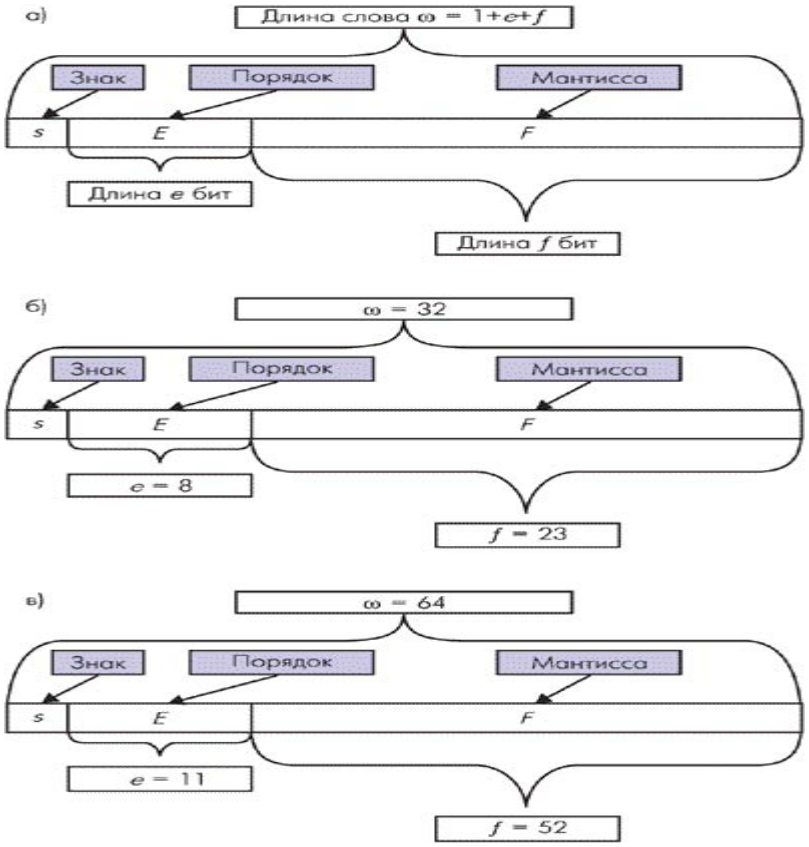


Рис. 32. Представление числа в формате с плавающей точкой:
общее представление (а), а также форматы *single* (б) и *double* (в)

Работая с библиотекой *Filter Design* и, в частности, с программой *fdatool*, пользователь может применять один из трёх форматов с плавающей точкой: *single, double* (с обычной и двойной точностью соответст-

венно), определяемые стандартом *IEEE* 754 для двоичной арифметики с плавающей точкой, а также формат *float*. Выбор того или иного формата осуществляется, как ранее было отмечено, в окнах колонки *Mode* страницы *Set Quantization Parameters*. В колонке *Format* устанавливается спецификация формата, имеющая ту же форму, что и для арифметики с фиксированной точкой: *[w, f]*, где *w* - длина слова, *f* - длина мантиссы.

В соответствии со стандартом *IEEE* 754 показатель записывается со смещением *B*. Это значит, что для получения истинного значения порядка надо из содержимого поля порядка, то есть из величины *E* вычесть величину *B*, определяемую по формуле $B = 2^{e-1} - 1$.

Например, если длина порядка равна *8 (e = 8)*, то *E* может принимать значения в интервале [0, 255]. Значит, значение порядка лежит в пределах интервала $[0, 255] - (2^{8-1} - 1) = [0, 255] - 127 = [-127, 128]$.

В системе *MATLAB* граничные значения порядка зарезервированы для особых случаев, например, для отображения результатов деления конечной величины на нуль *(inf)*, деления нуля на нуль *(NaN)* и др. Поэтому в приведённом примере диапазон порядка лежит в интервале [-126, 127].

Мантиссу числа с плавающей точкой можно записать по-разному. Сдвигая позицию двоичной точки влево или вправо, надо лишь соответствующим образом увеличивать или уменьшать порядок. Для достижения наибольшей точности мантиссу логично было бы расположить так, чтобы её старший бит (ближайший справа от точки) содержал единицу. Однако раз при таком расположении содержимое старшего бита мантиссы всегда единица, её нет нужды запоминать. Этот бит является скрытым. Следовательно, если под мантиссу отведено *e* бит, запоминается на самом деле *e*+1 бит (*e* бит, следующие за старшим, скрытым битом мантиссы, плюс скрытый бит). Таким образом, взаимосвязь между величиной, записанной в формате, показанном на рис. 32, *а*, и истинным значением *v* задаётся в виде $v = (-1)^s(2^{E-B})(1.F)$. Числа, представленные таким способом, называются нормализованными.

В некоторых случаях может возникнуть ситуация, когда результатом какой-либо арифметической операции является число, имеющее такое маленькое абсолютное значение, что для него величина *E* = порядок + $B < 0$.

Это явление называется исчезновением разрядов порядка, или антипереполнением порядка. Для того чтобы увеличить порядок и тем самым избежать явления переполнения, используются денормализованные числа. Они получаются из нормализованного представления сдвигом мантиссы вправо и соответствующим увеличением порядка (то есть уменьшением модуля порядка). Для денормализованных чисел взаимосвязь записи в формате, показанном на рис. 31, *а*, и истинного значения определяется соотношением $v = (-1)^s(2^{-B+1})(0.F)$.

Читатель может заметить, что для задания знака числа с плавающей точкой отведён один бит, в то время как знак имеют и мантисса, и порядок. В описываемых форматах знаковый бит содержит знак мантиссы, знак же порядка определяется по результату вычитания смещения *B* из содержимого поля порядка *E*.

Представления чисел при использовании форматов *single* и *double* показаны на рис 32, *б*, *в* соответственно. При выборе одного из этих форматов автоматически устанавливаются параметры в колонках *Round mode, Overflow mode* и *Format.* Параметрами *Round mode* в этом случае являются значения *round*, а *Overflow mode - saturate*. В табл. 3 указаны допустимые значения порядков и численных значений при использовании форматов *single* и *double.*

Ещё один формат, с которым работает *fdatool* - это *float*. Представление чисел в этом формате также соответствует приведённым выше формулам и обозначениям, показанным на рис. 32, *а*, но в этом случае пользователь может произвольно задавать длины мантиссы и порядка, а также устанавливать значения в полях *Format* и *Round mode*.

Допустимые значения порядков и численных значений при использовании форматов *single* и *double* приведены в табл. 3.

Таблица 3

Формат записи и хранения числа	*Single*		*double*	
	Порядок	Значение	Порядок	Значение
Нормализованные числа	$0 < E < 255$	$v = (-1)^s(2^{E-127})(1.F)$	$0 < E < 2047$	$v = (-1)^s(2^{E-1023})(1.F)$
Денормализованные числа	$E = 0$	$v = (-1)^s(2^{-126})(0.F)$	$E = 0$	$v = (-1)^s(2^{-1022})(0.F)$

При этом длина порядка должна находиться в интервале *1 ⩽e ⩽11,* а длина слова - в интервале *e+1 ⩽w ⩽64.* Независимо от выбранного формата *(single, double или float*) в случае возникновения переполнения на выходе соответствующего квантователя устанавливается значение *inf* или *-inf*, поэтому окна колонки *Overflow mode* неактивны.

Проектирование цифрового фильтра: квантование параметров и анализ результатов

Теперь после знакомства с основными причинами, порождающими эффекты квантования, вновь обратимся к нашей задаче проектирования цифрового фильтра (рис. 33). Ясно, что различие между АЧХ фильтра-прототипа и АЧХ квантованного фильтра при выбранной структуре связано с установками параметров квантования. В данном случае использовались форматы с фиксированной точкой, что соответствует значению *fixed* в поле *Mode* и значениям для квантователей *Coefficient, Input, Output* и *Multiplicand* в поле *Format.* Для квантователей *Product* и *Sum* используемый формат.

Quantized Direct form II transposed filter
Numerator

	QuantizedCoefficients{1}	*ReferenceCoefficients{1}*
(1)	*0.000335693359375*	*0.000330315544487904*
(2)	*-0.001007080078125*	*-0.001020081231375195*
(3)	*0.002593994140625*	*0.002580261271031570*
(4)	*-0.004577636718750*	*-0.004580740133027181*
(5)	*0.007354736328125*	*0.007347725797754056*
(6)	*-0.009704589843750*	*-0.009715041792670265*
(7)	*0.011993408203125*	*0.011986955602516003*
(8)	*-0.013122558593750*	*-0.013120022874790487*
(9)	*0.013793945312500*	*0.013803049933740345*
(10)	*-0.013122558593750*	*-0.013120022874790454*
(11)	*0.011993408203125*	*0.011986955602515940*
(12)	*-0.009704589843750*	*-0.009715041792670199*
(13)	*0.007354736328125*	*0.007347725797753998*
(14)	*-0.004577636718750*	*-0.004580740133027142*
(15)	*0.002593994140625*	*0.002580261271031547*

Рис. 33. Параметры фильтра-прототипа и квантованного фильтра (см. также с. 54)

(16) -0.001007080078125 -0.001020081231375187
(17) 0.000335693359375 0.000330315544487902
Denominator
QuantizedCoefficients{2} ReferenceCoefficients{2}

+(1) 0.999969482421875 1
-(2) -1.000000000000000 -4.0016106715603197
+(3) 0.999969482421875 13.979587570568736
-(4) -1.000000000000000 -31.703958726815337
+(5) 0.999969482421875 63.654203987427906
-(6) -1.000000000000000 -100.21434456606917
+(7) 0.999969482421875 141.70189019442469
-(8) -1.000000000000000 -166.01889747992664
+(9) 0.999969482421875 175.86734947768485
-(10) -1.000000000000000 -156.48177861802839
+(11) 0.999969482421875 125.88638778866985
-(12) -1.000000000000000 -83.883343560153804
+(13) 0.999969482421875 50.201260500070624
-(14) -1.000000000000000 -23.538510095548396

-(16) -1.000000000000000 -2.6300444174959825
(17) 0.619323730468750 0.61932234850652279
FilterStructure=df2t
ScaleValues= [1]
NumberOfSections=1
StatesPerSection= [16]
CoefficientFormat = quantizer(“fixed”, “round”, “saturate”, [16 15])
InputFormat = quantizer(“fixed”, “floor”, “saturate”, [16 15])
OutputFormat = quantizer(“fixed”, “floor”, “saturate”, [16 15])
MultiplicandFormat = quantizer(“fixed”, “floor”, “saturate”, [16 15])
ProductFormat = quantizer(“fixed”, “floor”, “saturate”, [32 30])
SumFormat = quantizer(“fixed”, “floor”, “saturate”, [32 30])
Warning: 16 overflows in coefficients.

Рис. 33. Окончание

Для того чтобы сравнить коэффициенты фильтров, нажмём кнопку "Коэффициенты фильтра". В области отображения параметров распечатается следующая информация, показанная на рис. 33. Как видно из распечатки, все коэффициенты знаменателя передаточной функции фильтра-прототипа за исключением последнего превышают

по модулю единицу, в то время как установленный формат для квантователя *Coefficient* равен. Это означает, что единственный бит, не предназначенный для хранения дробной части числа, является знаковым, то есть квантование любого числа, модуль которого больше или равен единице, приведёт к переполнению. Из-за эффекта переполнения коэффициенты знаменателя квантованного фильтра существенно отличаются от соответствующих коэффициентов фильтра-прототипа, что отмечено в начале каждой строки, за исключением последней, знаком "+" (переполнение в сторону плюс бесконечности) или "–" (переполнение в сторону минус бесконечности).

Попробуем промасштабировать коэффициенты так, чтобы они по модулю не превышали единицу. Это необходимо для повышения точности расчётов при реализации фильтров на *DSP* и осуществляется нажатием кнопки *Scale transferfcn coeffs <= 1*, расположенной в левой верхней части страницы *Set Quantization Parameters*. Для масштабирования используются коэффициенты, равные степени 2. После выполнения расчётов в поле *Input/output scaling*, расположенном под указанной кнопкой, появятся значения масштабных множителей, на которые умножаются входной и выходной сигналы фильтра с изменёнными коэффициентами. АЧХ фильтров будут выглядеть так, как показано на рис. 34.

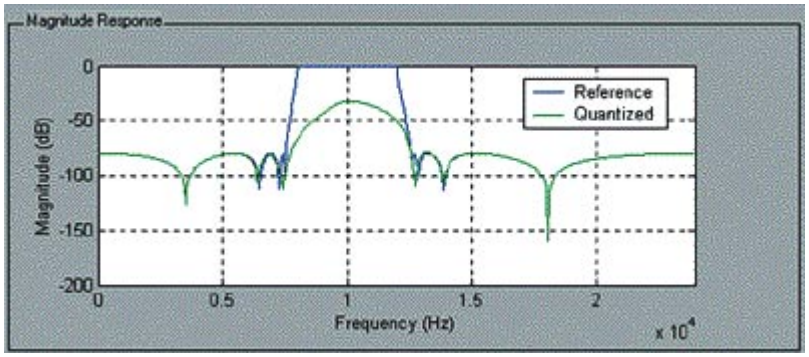


Рис. 34. Амплитудно-частотные характеристики фильтров после масштабирования коэффициентов

Помимо искажений АЧХ в результате квантования появляются шумы, фильтр может стать неустойчивым, и возникнут разные виды генерации (например генерация периодических колебаний, предель-

ные циклы высокого и низкого уровней), что в принципе затрудняет определение АЧХ. *fdatool* не позволяет решать задачи расчёта шумов арифметики, расчёта предельных циклов, оптимизации динамического диапазона и ряд других. Для некоторых из этих целей в *MATLAB* имеются специальные функции (например *nlm, limitcycle* и другие). Однако исключительно важное значение *fdatool* состоит в том, что он позволяет создавать нелинейные модели цифровых фильтров, весьма точно отражающих динамику реальных систем. Эти модели и являются объектами для упомянутых выше функций.

Возвращаясь к нашему примеру, щёлкнем мышью, поместив предварительно курсор в редактируемое окно *Input/output scaling*, нажмём на клавиатуре *Enter* и затем клавишу *Apply*, находящуюся внизу главного окна *fdatool*. В области *Current Filter Information* появится сообщение *"Stable: No"*. В том, что фильтр неустойчив, можно также убедиться, нажав на кнопку *Pole/Zero Plot* (на рис. 34 она обозначена как "Полюсы и нули"). В результате появится карта нулей и полюсов, показанная на рис. 35, на которой видно, что передаточная функция построенного квантованного фильтра имеет полюсы, лежащие вне единичного круга.

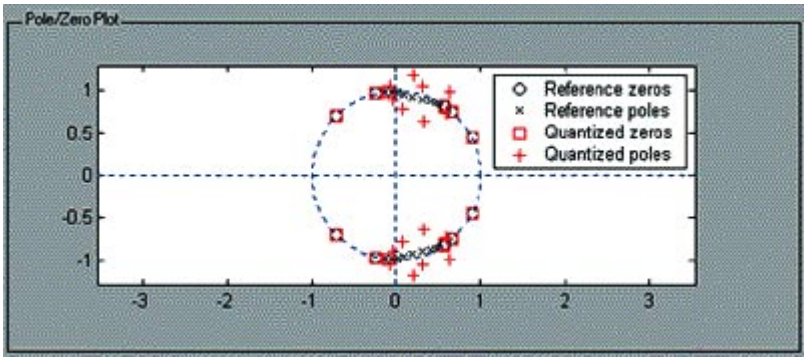


Рис. 35. Карта нулей и полюсов

Теперь попробуем решить поставленную задачу иначе, но прежде восстановим наши результаты, записанные на диск. Для этого выберем раздел *Open Session* меню *File* и откроем файл. В области *Current Filter Information* нажмём клавишу *Convert Structure…* и в открывшемся окне установим флажок *Use second order sections* (рис. 36), указав тем самым, что проектируемый квантованный фильтр должен быть реализован звеньями второго порядка. Нажатие кнопки *Apply*

или *OK* инициализирует расчёт фильтра в указанной реализации. После вычислений можно убедиться, что АЧХ фильтра-прототипа и квантованного фильтра практически совпадают. Полученный результат показан на рис. 37. Важно обратить внимание на содержимое окна *Input/output scaling*: в нём распечатаны масштабирующие множители для сигналов на входе каждого из звеньев второго порядка, а также для выходного сигнала последнего звена.

Итак, на первый взгляд, поставленная задача решена - цифровой фильтр с требуемыми характеристиками построен. Однако следует заметить, что частотная характеристика фильтра и, в частности АЧХ, показанная на рис. 37, построена по рассчитанным квантованным коэффициентам в предположении, что фильтр является идеальной линейной системой. Другими словами, при расчёте частотной характеристики учитывались лишь эффекты квантования коэффициентов фильтра. Для получения реальной частотной характеристики необходимо учесть нелинейные эффекты квантования, связанные с формой реализации (структурой) фильтра и квантованием других параметров, устанавливаемых на странице *Set Quantization Parameters*.

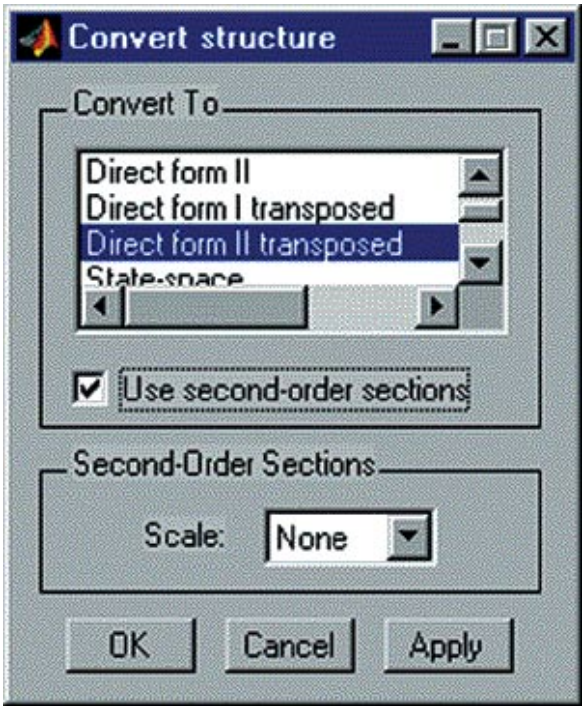


Рис. 36. Установка флажка Use second order sections в окне Convert Structure

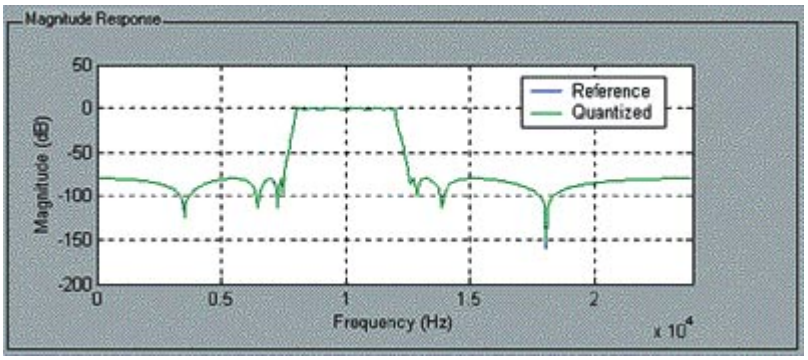


Рис. 37. Амплитудно-частотные характеристики фильтра-прототипа и квантованного фильтра, реализованного звеньями второго порядка

Чтобы получить частотную характеристику фильтра, построенную с учётом указанных факторов, следует выполнить комплексную проверку его работы на основе имитационного моделирования. Рассмотрим, как можно определить АЧХ, вообще говоря, нелинейной системы, используя одну из функций *MATLAB - nlm (noise loading method)*, которая вычисляет оценку частотной характеристики квантованного фильтра с учётом всех эффектов квантования. Функция на основе метода Монте-Карло выполняет *L* испытаний, в каждом из которых моделируется прохождение входного сигнала, представляющего собой белый шум, через исследуемый фильтр. По результатам каждого испытания находится оценка частотной характеристики фильтра по спектрам входного и выходного сигналов. Итоговая оценка частотной характеристики вычисляется усреднением оценок частотных характеристик по всем испытаниям.

Перед тем как воспользоваться функцией *nlm*, экспортируем наш фильтр в рабочее пространство *MATLAB (Work-space)*. Это выполняется следующим образом. В меню *File* выбрать раздел *Export*, в появившемся окне указать, куда экспортировать *(Workspace*) и имя фильтра (например *Hq1*), после чего нажать кнопку *OK*. После выполнения команды надо перейти в командное окно *MATLAB* и набрать команду *>> nlm(Hq1,[],50),* по которой будет выполнено моделирование, включающее 50 испытаний. В результате появится окно с графиками АЧХ и ФЧХ, из которых видно, что построенная нами нелинейная цифровая система, каковым является квантованный фильтр, не удовлетворяет заданным требованиям, изложенным в

табл. 1 и показанным на рис. 38. Отметим, что функцию *nlm* можно вызвать, указав лишь имя фильтра >> *nlm(Hq1)*.

В этом случае число испытаний L будет выбрано по умолчанию ($L = 10$), и получаемые графики будут менее сглаженными.

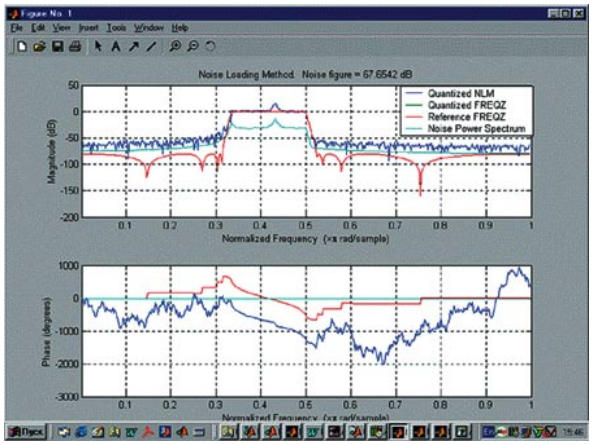


Рис. 38. Результаты моделирования с помощью функции *nlm*

Очевидно, что для получения желаемого результата нам надо изменить параметры квантования, задаваемые на странице *Set Quantization Parameters*. Попробуем для всех квантователей, кроме квантователей коэффициентов, переустановить способ округления с *floor* в *convergent* и рассчитать новый квантованный фильтр, нажав кнопку *Apply*. Если затем импортировать полученный фильтр в *Workspace* и повторить вызов функции *nlm* к этому фильтру, получим частотную характеристику. Сравнивая графики, изображенные на рис. 38 и 39, видно, что АЧХ последнего фильтра является лучшим приближением к требуемому результату, однако она тоже не удовлетворяет заданным требованиям.

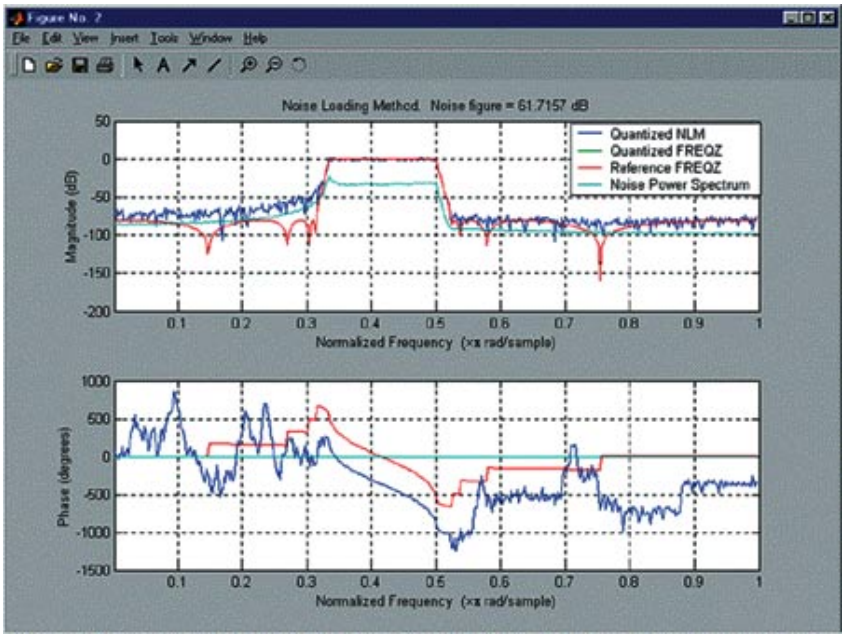


Рис. 39. Результаты моделирования с помощью функции *nlm* после замены способа округления с *floor* на *convergent*

Обратимся снова к странице *Set Quantization Parameters* и установим форматы квантователей в соответствии с рис. 40. Повторив процедуру расчёта, импорта и испытания фильтра с помощью *nlm*, получим результат, показанный на рис. 41. Как видно из рисунка, АЧХ фильтра-прототипа и квантованного фильтра практически совпадают, то есть полученный фильтр удовлетворяет требованиям.

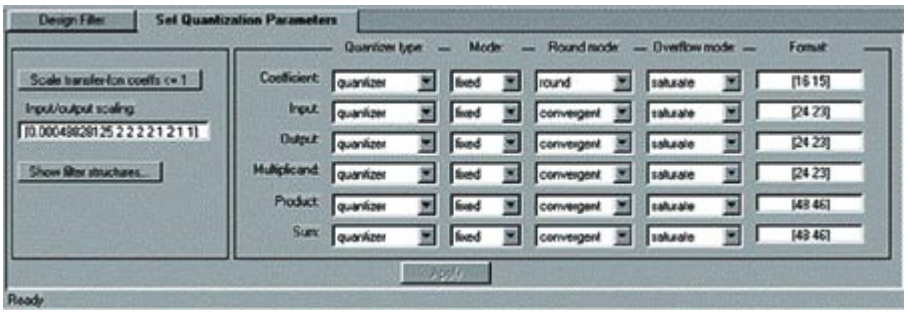


Рис. 40. Изменение форматов квантователей

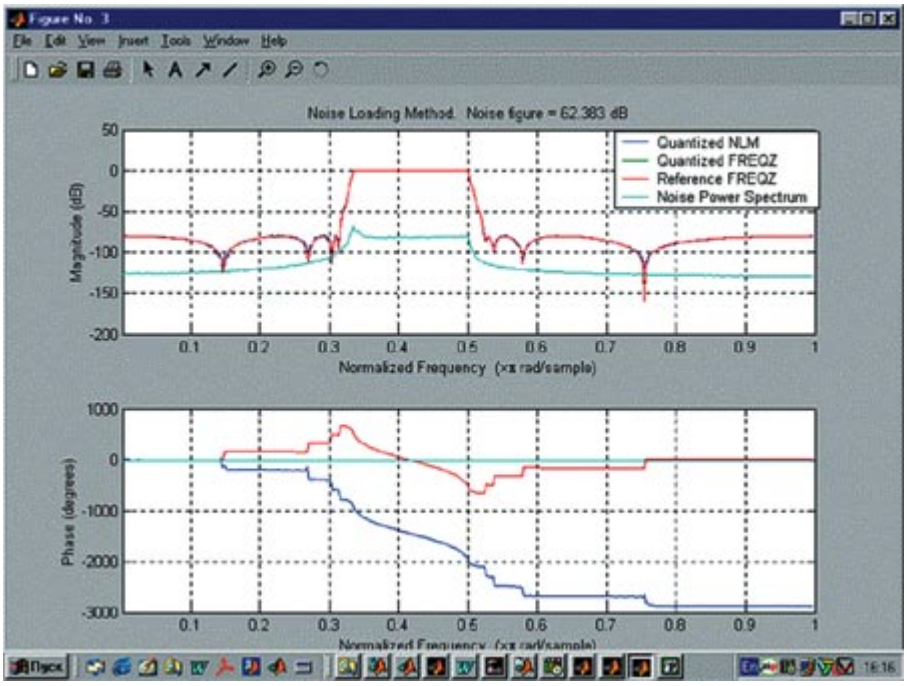


Рис. 41. Результаты моделирования с помощью функции *nlm* после замены форматов квантователей

Порядок выполнения работы

1. Спроектировать цифровой эллиптический полосовой фильтр, требования к которому заданы. Для этого нам необходимо выделить и заполнить соответствующие позиции на странице *Filter Design* глав-

ного окна *fdatool*, как показано на рис. 42, после чего нажать на клавишу *Design Filter*, расположенную внизу окна. После завершения расчёта в верхней части окна будут представлены результаты.

2. В главном окне ниже раздела *Current Filter Information* находится раздел *Quantization*, содержащий единственное поле *Turn Quantization On.* Для запуска процедуры расчёта установим в этом поле флажок, запустив тем самым процедуру расчёта квантованного фильтра с параметрами квантования, заданными по умолчанию. По окончании процедуры расчёта АЧХ нового (квантованного) фильтра наложится на АЧХ фильтра-прототипа, рассчитанного ранее. Сохраним полученные результаты на диске, для чего откроем меню *File* и выберем раздел *Save Session As* под именем *HQ1*…

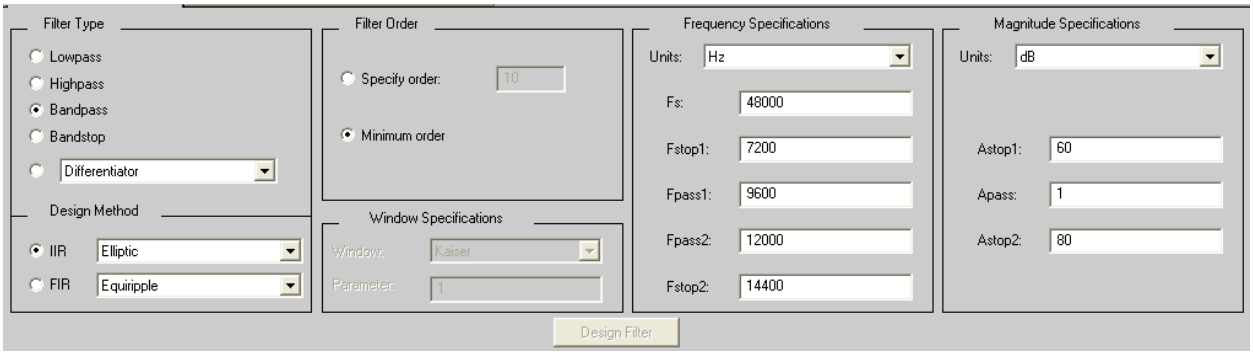


Рис. 42. Главное окно fdatool на странице Filter Design

3. Для того чтобы сравнить коэффициенты фильтров, нажмём кнопку "Коэффициенты фильтра". Попробуем промасштабировать коэффициенты так, чтобы они по модулю не превышали единицу. Это необходимо для повышения точности расчётов при реализации фильтров на *DSP* и осуществляется нажатием кнопки *Scale transferfcn coeffs <= 1*, расположенной в левой верхней части страницы *Set Quantization Parameters*. Для масштабирования используются коэффициенты, равные степени 2. После выполнения расчётов в поле *Input/output scaling*, расположенном под указанной кнопкой, появятся значения масштабных множителей, на которые умножаются входной и выходной сигналы фильтра с изменёнными коэффициентами. Помимо искажений АЧХ в результате квантования появляются шумы, фильтр может стать неустойчивым и возникнут разные виды генера-

ции (например генерация периодических колебаний, предельные циклы высокого и низкого уровней), что в принципе затрудняет определение АЧХ. *fdatool* не позволяет решать задачи расчёта шумов предельных циклов, оптимизации динамического диапазона и ряд других. Для некоторых из этих целей в *MATLAB* имеются специальные функции (например *nlm, limitcycle* и другие). Однако исключительно важное значение *fdatool* состоит в том, что он позволяет создавать нелинейные модели цифровых фильтров, весьма точно отражающих динамику реальных систем. Эти модели и являются объектами для упомянутых выше функций.

4. Щёлкнув мышью, поместить предварительно курсор в редактируемое окно *Input/output scaling*, нажмём на клавиатуре *Enter*, и затем нажмём клавишу *Apply*, находящуюся внизу главного окна *fdatool*. В области *Current Filter Information* появится сообщение *"Stable: No"*. В том, что фильтр неустойчив, можно также убедиться, нажав на кнопку *Pole/Zero Plot.*

Теперь попробуем решить поставленную задачу иначе, но прежде восстановим наши результаты, записанные на диск. Для этого выберем раздел *Open Session* меню *File* и откроем файл. В области *Current Filter Information* нажмём клавишу *Convert Structure…* и в открывшемся окне установим флажок *Use second order sections*, указав тем самым, что проектируемый квантованный фильтр должен быть реализован звеньями второго порядка. Нажатие кнопки *Apply* или *OK* инициализирует расчёт фильтра в указанной реализации. После вычислений можно убедиться, что АЧХ фильтра-прототипа и квантованного фильтра практически совпадают. Важно обратить внимание на содержимое окна *Input/output scaling*: в нём распечатаны масштабирующие множители для сигналов на входе каждого из звеньев второго порядка, а также для выходного сигнала последнего звена.

Итак, на первый взгляд задача, поставленная нами (см. рис. 42), решена - цифровой фильтр с требуемыми характеристиками построен. Однако следует заметить, что частотная характеристика фильтра и, в частности АЧХ, построена по рассчитанным квантованным коэффициентам в предположении, что фильтр является идеальной линейной

системой. Другими словами, при расчёте частотной характеристики учитывались лишь эффекты квантования коэффициентов фильтра. Для получения реальной частотной характеристики необходимо учесть нелинейные эффекты квантования, связанные с формой реализации (структурой) фильтра и квантованием других параметров, устанавливаемых на странице *Set Quantization Parameters.*

Чтобы получить частотную характеристику фильтра, построенную с учётом указанных факторов, следует выполнить комплексную проверку его работы на основе имитационного моделирования. Рассмотрим, как можно определить АЧХ, вообще говоря, нелинейной системы, используя одну из функций *MATLAB - nlm (noise loading method)*, которая вычисляет оценку частотной характеристики квантованного фильтра с учётом всех эффектов квантования. Функция на основе метода Монте-Карло выполняет *L* испытаний, в каждом из которых моделируется прохождение входного сигнала, представляющего собой белый шум, через исследуемый фильтр. По результатам каждого испытания находится оценка частотной характеристики фильтра по спектрам входного и выходного сигналов. Итоговая оценка частотной характеристики вычисляется усреднением оценок частотных характеристик по всем испытаниям.

5. Перед тем как воспользоваться функцией *nlm,* экспортируем наш фильтр в рабочее пространство *MATLAB (Work-space)*. Это выполняется следующим образом. В меню *File* выбрать раздел *Export*, в появившемся окне указать, куда экспортировать (*Workspace*) и имя фильтра (например *Hq1*), после чего нажать кнопку *OK*. После выполнения команды надо перейти в командное окно *MATLAB* и набрать команду *>> nlm(Hq1,[],50),* по которой будет выполнено моделирование, включающее 50 испытаний. В результате появится окно с графиками АЧХ и ФЧХ, из которых видно, что построенная нами нелинейная цифровая система, каковым является квантованный фильтр, не удовлетворяет заданным требованиям.

Очевидно, что для получения желаемого результата надо изменить параметры квантования, задаваемые на странице *Set Quantization Parameters*. Попробуем для всех квантователей, кроме квантователей

коэффициентов, переустановить способ округления с *floor* в *convergent* и рассчитать новый квантованный фильтр, нажав кнопку *Apply*. Если затем импортировать полученный фильтр в *Workspace* и повторить вызов функции *nlm* к этому фильтру, получим частотную характеристику. Сравнивая графики, видно, что АЧХ последнего фильтра является лучшим приближением к требуемому результату, однако она тоже не удовлетворяет заданным требованиям.

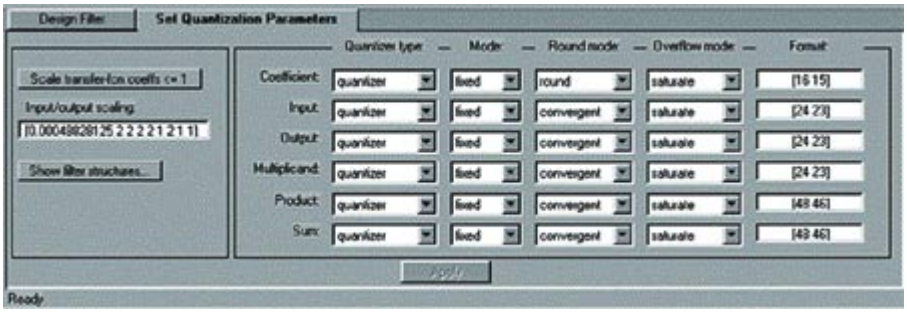


Рис. 43. Изменение форматов квантователей

Обратимся снова к странице *Set Quantization Parameters* и установим форматы квантователей в соответствии с рис. 43. Повторив процедуру расчёта, импорта и испытания фильтра с помощью *nlm*, получим результат. Как видно из рисунка, АЧХ фильтра-прототипа и квантованного фильтра практически совпадают, то есть полученный фильтр удовлетворяет требованиям, изложенным в табл. 1.

Содержание отчета

Отчет должен содержать

1. Графики амплитудно-частотной характеристики фильтра-прототипа.

2. Графики амплитудно-частотной характеристики фильтров после масштабирования коэффициентов.

Контрольные вопросы

1. Назовите основные принципы реализации цифровых фильтров?

2. Что представляет собой амплитудно-частотные характеристики фильтров после масштабирования коэффициентов?

Лабораторная работа № 8

ПРЕОБРАЗОВАНИЕ СИГНАЛОВ В ЦИФРОВЫХ ФИЛЬТРАХ

Цель работы: изучить, как преобразуются сигналы и их спектры при прохождении сигналов через цифровые фильтры. В работе используется программная среда *MATLAB* и её программный пакет *Signal Processing*.

Варианты заданий приведены в табл. 4.

Таблица 4

Вариант	Сигнал	Фильтр
1	Прямоугольный импульс	ФНЧ Баттерворта, Чебышева I
2	Треугольный импульс	ФНЧ Баттерворта, Чебышева II
3	Синусоидальный импульс	ФНЧ Баттерворта, эллиптический
4	Прямоугольный радиоимпульс	ППФ Баттерворта, Чебышева I
5	Треугольный радиоимпульс	ППФ Баттерворта, Чебышева II
6	Синусоидальный радиоимпульс	ППФ Баттерворта, эллиптический

Теоретическая часть

Цифровой сигнал задаётся в виде последовательности отсчётов, выраженных числами в двоичном коде. Отсчёты привязаны к моментам дискретного времени $n = 0,1,2\ldots$. Спектр сигнала можно найти, применив к нему дискретное преобразование Фурье (ДПФ):

$$S[k]=\sum_{n=0}^{N-1} s[n]\ exp(-j2\pi nk/N),\ k=0,1,\ldots N-1,$$

где N – количество отсчётов сигнала. Дискретное время n соответствует реальному времени $t=nT$, где T – интервал дискретизации, выраженный в секундах. Дискретная частота k может быть поставлена в

соответствие реальной частоте $f=kFs/N$, *где* $Fs=1/T$ – частота дискретизации в герцах. Частота дискретизации выбирается исходя из условия теоремы Котельникова: $Fs \geq 2Fm$, где Fm – максимальная частота в спектре сигнала. По смыслу ДПФ сигнал $s[n]$ периодически продолжается с периодом N и с тем же периодом будет продолжен спектр $S[k]$ в области дискретной частоты.

Сигнал может быть пропущен через цифровой фильтр (ЦФ), то есть профильтрован. Фильтрация во временной области выполняется путём вычисления отсчётов выходного сигнала по разностным уравнениям, соответствующим функциональной схеме ЦФ. Фильтр в этом случае характеризуется набором коэффициентов bk и ak передаточной функции $K(z)$. При фильтрации в зависимости от формы АЧХ и ФЧХ фильтра изменяются соотношения между амплитудами спектральных составляющих сигнала, а также становятся другими начальные фазы этих составляющих. Изменение спектра приводит и к соответствующему изменению формы сигнала. Если существенная часть спектра сигнала укладывается в полосу пропускания фильтра, то искажения формы сигнала будут незначительными. В противном случае форма сигнала может довольно сильно измениться.

Порядок выполнения работы

1. Получив вариант задания, изобразите в рабочей тетради график сигнала, обозначьте на нём наименования параметров, характеризующих сигнал (амплитуда A, длительность τ, интервал следования импульсов T, частота несущего колебания $f0$). Запишите тип и классы фильтров для вашего варианта задания.

2. Задайте численные значения параметров (в разумных пределах). Вспомните, каким образом оценивается ширина спектра сигнала исходя из его длительности, и рассчитайте примерную ширину спектра. Не забудьте, что при выборе частоты несущего колебания радиоимпульсов нужно соблюсти условие узкополосности ($\Delta fcn << f0$). Выберите частоту дискретизации Fs исходя из условия теоремы Котельникова ($Fs \geq 2Fm$, где Fm – максимальная частота в спектре

сигнала). Учтите, что спектры сигналов, содержащих скачки (прямоугольный импульс, экспоненциальный импульс), убывают с ростом частоты не очень резко, поэтому для лучшего представления сигнала стоит выбрать увеличенное значение частоты дискретизации. В качестве проверки рассчитайте, сколькими отсчётами будет представлен видеоимпульс*:* $Nomc=\tau \cdot Fs$, где τ -длительность импульса (для экспоненциального импульса можно взять $\tau = 5\ \tau 0$, где $\tau 0$ – постоянная времени экспоненты); *Nomc* должно составлять примерно 10…20. Для радиоимпульсов период несущего колебания должен быть представлен 8…10 отсчётами.)

3. Загрузите *MATLAB*. В командном окне задайте частоту дискретизации, интервал времени, в течение которого будет длиться сигнал (учитывается время, отводимое на импульсы и на паузы между ними), и сформируйте вектор отсчётов сигнала, руководствуясь информацией, приведённой ниже.

Сигнал задаётся в виде вектора, сопоставленного с вектором моментов времени. Перед вводом модели сигнала нужно указать частоту дискретизации и сформировать вектор-столбец моментов времени. Например,

>> Fs = 1e3; t=0:1/Fs:1; t=t';

В данном случае введена частота дискретизации 1кГц. Сигнал будет задан на интервале времени 1с (1001 отсчёт). Последний оператор означает преобразование вектора-строки в вектор-столбец (‘ – операция транспонирования матрицы). Не следует забывать ставить точку с запятой в конце каждого оператора, чтобы подавить вывод значений на экран монитора.

Рассмотрим некоторые из возможных сигналов.

а) Прямоугольный импульс (рис. 44)

>>A=5; tau=5e-2;

>> Fs = 1.5e3; t=0:1/Fs:1; t=t';

>> s=A · rectpuls (t – tau/2, tau);

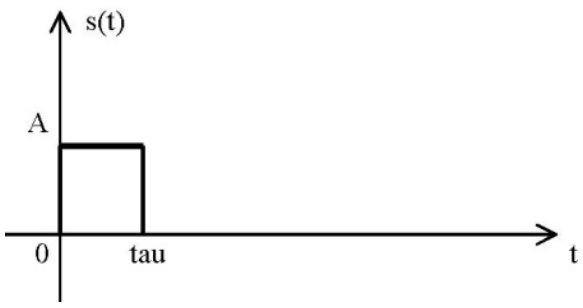


Рис. 44. График сигнала прямоугольного импульса

При вводе этого оператора либо нужно предварительно задать значения амплитуды *A* и длительности *tau*, либо в самом операторе вместо идентификаторов *A* и *tau* поставить численные значения.

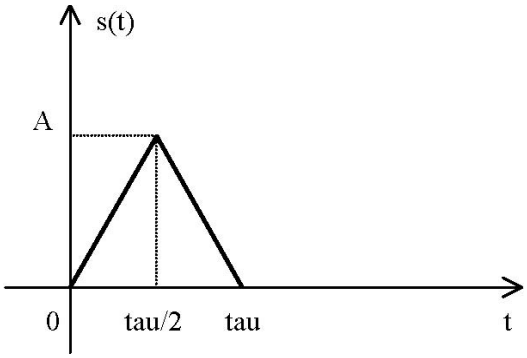


Рис. 45. График сигнала треугольного импульса

б) Треугольный импульс (рис. 45)

>>A=5; tau=7e-2;

>> Fs = 1.5e3; t=0:1/Fs:1e-1; t=t';

>> s= A · tripuls (t – tau/2, tau);

в) Экспоненциальный импульс (рис. 46)

>> s = A · exp (- t / tau).

Подразумевается, что вектор *t* задан для моментов времени $t \geq 0$.

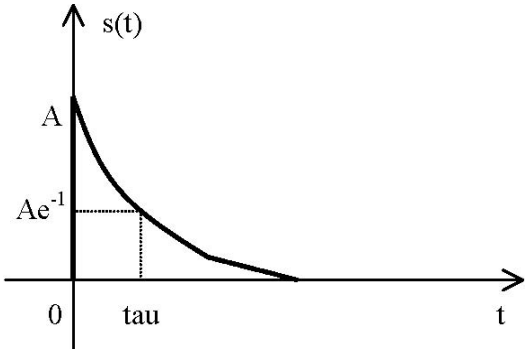


Рис. 46. График сигнала экспоненциального импульса

г) Синусоидальный импульс (рис. 47)

>>A=5; tau=7e-2;

>> Fs = 1.5e3; t=0:1/Fs:1e-1; t=t';

>> s = A · sin (pi · t / tau) . (t>=0) · (t<=tau).*

Здесь используется тот факт, что операции сравнения возвращают 1, если неравенство выполняется, или 0 в противном случае.

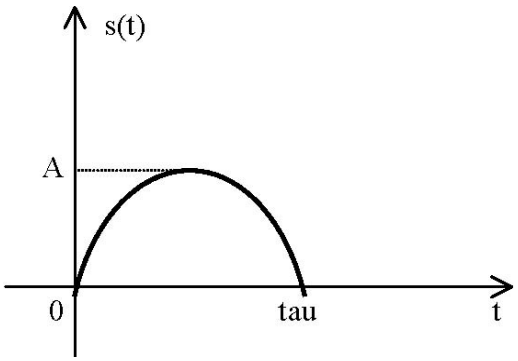


Рис. 47. График сигнала синусоидального импульса

д) Радиоимпульсы

Они получаются при умножении видеоимпульса *s* на гармоническое колебание

>> s1 = s · cos (2 · pi · f0 · t + phi).

Предварительно нужно задать значение несущей частоты *f0* и начальной фазы *phi*. Обратите внимание, что операция умножения представлена здесь как .* (точка перед знаком *). Это означает поэлементное умножение векторов (в противном случае производилась бы операция матричного умножения). В тех случаях,

когда осуществляется умножение скаляров или матрицы (вектора) на скаляр, можно использовать символ *. То же самое относится к операции деления и возведения в степень. Поэлементное деление матриц задаётся оператором . /, поэлементное возведение в степень . ^. Число π задаётся в *MatLab* как *pi*.

Прямоугольный радиоимпульс:

>>A=5; tau=5e-2;

>> Fs = 1.5e3; t=0:1/Fs:1; t=t';

>> s=A · rectpuls (t – tau/2, tau);

>> f0=4000;phi=pi/2;

>> s1 = s · cos (2 · pi · f0 · t + phi).

Треугольный радиоимпульс:

>>A=5; tau=7e-2;

>> Fs = 1.5e3; t=0:1/Fs:1e-1; t=t';

>> s= A · tripuls (t – tau/2, tau);

>> f0=4000;phi=pi/2;

>> s1 = s · cos (2 · pi · f0 · t + phi).

Синусоидальный радиоимпульс:

>>A=5; tau=7e-2;

>> Fs = 1.5e3; t=0:1/Fs:1e-1; t=t';

>> s = A · sin (pi · t / tau) · (t>=0) · (t<=tau);

>> f0=4000;phi=pi/2;

>> s1 = s · cos (2 · pi · f0 · t + phi).

е) Последовательность импульсов

Для генерации конечной последовательности (пачки) импульсов одинаковой формы с произвольно задаваемыми задержками и амплитудами используется функция *pulstran.* Она вызывается следующим образом:

s= pulstran (t, d, 'func', p1, p2 ...).

Здесь *t* - вектор значений моментов времени, *d* - вектор задержек и амплитуд импульсов, '*func*' - имя функции, задающей одиночный импульс, например, '*rectpuls*' или '*tripuls*'; *p1, p2 …* - параметры одиночного импульса, передаваемые функции *func*.

Например, нужно задать следующую последовательность прямоугольных импульсов (рис. 48).

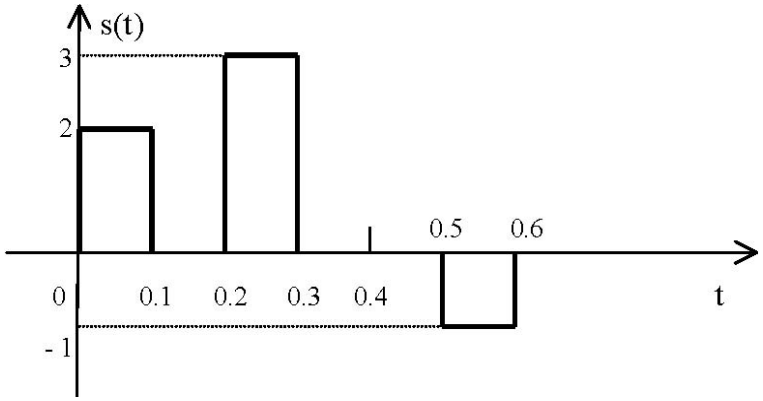


Рис. 48. График последовательности прямоугольных импульсов

Вводится набор операторов:

>> Fs= 1e3; t= 0:1/Fs:1; t= t';

>> tau= 0.1;

>> d(: ,1)= [0.05 0.25 0.55]' ;

>> d(: ,2)= [2 3 - 1]';

>> s= pulstran (t, d, 'rectpuls', tau).

Если нужно построить последовательность импульсов произвольной формы, причём отсчёты одиночного импульса записаны в векторе *s1*, то используют следующую форму задания функции *pulstran*:

s= pulstran (t, d, s1, Fs).

Например, нужно задать последовательность из четырёх синусоидальных импульсов (рис. 49). Вводятся следующие операторы:

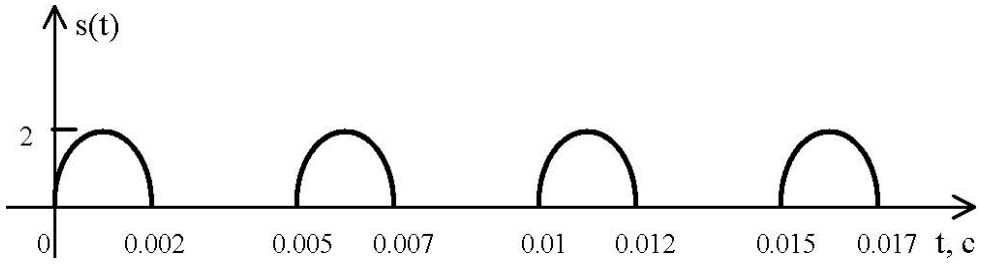


Рис. 49. График последовательности синусоидальных импульсов

>>Fs=1e4;
>>t=0:1/Fs:2e-2;t=t';
>>tau=2e-3;A=2;
*>>s1=sin(pi*t/tau).*(t<=tau);*
*>>d(:,1)=(0:3)'*5e-3;*
*>>d(:,2)=A*ones(4,1);*
>>s=pulstran(t,d,s1,Fs).

4. Вызовите программу *SPTool (Signal Processing Tool)*. Для этого нужно в командном окне *MATLAB* набрать *>> sptool.*

Импортируйте в *sptool* сформированный сигнал. С этой целью выберите команду *Import* в меню *File* главного окна программы *sptool*. Появится окно *Import to SPTool* (рис. 50). Переключатель *Source* установите в положение *From Workspace*. В списке *Workspace Contents* перечислены переменные, имеющиеся в рабочей области *MatLab* (для радиосигналов выбрать *s1*, для остальных - *s*). В раскрывающемся списке *Import As* нужно выбрать строку *Signal*. Далее выберите в списке идентификатор вектора, содержащего отсчёты сигнала, и нажмите кнопку --> напротив поля *Data.* Затем аналогичным образом введите значение частоты дискретизации *Fs* в поле *Sampling Frequency*. Задайте имя сигнала в поле *Name*. Под этим именем он будет помещён в список сигналов *sptool*. Вслед за этим следует нажать кнопку *OK*.

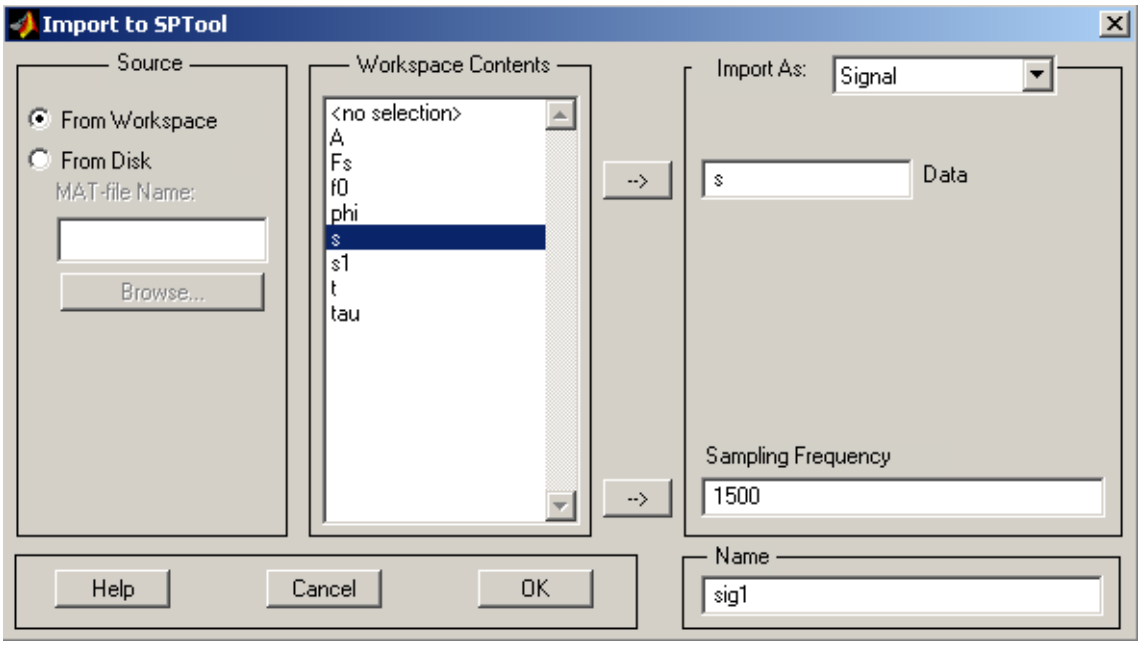


Рис. 50. Окно Import to SPTool

Просмотрите график сигнала. Для этого выделите имя сигнала в списке сигналов и нажмите кнопку *View*, расположенную под этим списком. При просмотре графиков можно изменять масштаб, укрупняя отдельные участки. Для измерения значений в отдельных точках используются маркеры, которые можно перетаскивать мышью.

5. Рассчитайте спектр сигнала. С этой целью выделите сигнал в списке сигналов и нажмите кнопку *Create*, расположенную под списком спектров. В окне *Spectrum Viewer* в поле *Parameters* нужно указать метод спектрального анализа. Укажите метод ДПФ (используется быстрое преобразование Фурье БПФ (*FFT*)). Указав метод, следует щёлкнуть мышью по кнопке *Apply*. Будет выведен график спектральной плотности мощности (рис. 51). Имеется возможность выводить спектры в линейном или в логарифмическом масштабе (меню *Options*). Зарисуйте спектр сигнала.

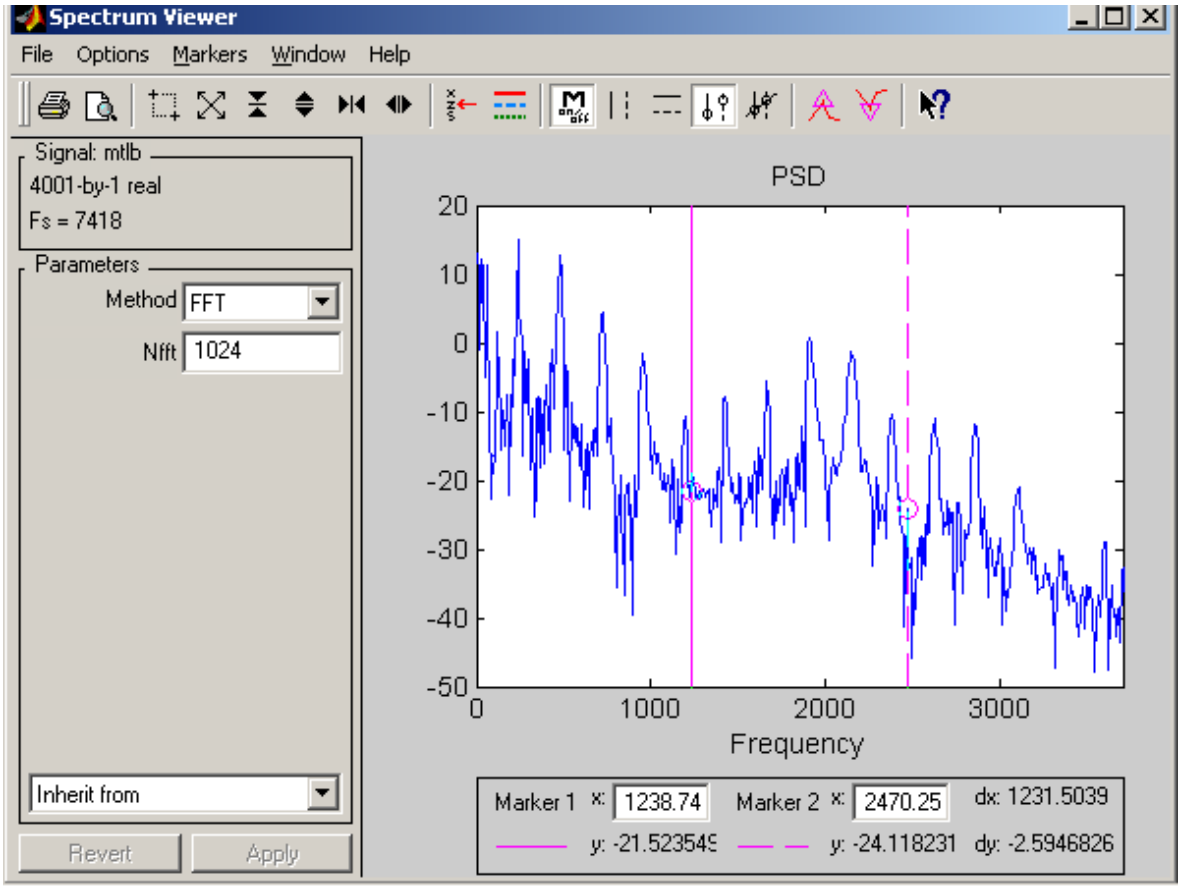


Рис. 51. График спектральной плотности мощности

6. Руководствуясь спектром сигнала, определите, каковы должны быть полосы пропускания и задерживания фильтра, чтобы пропустить нижние частоты спектра и подавить верхние или наоборот, или пропустить средние частоты и подавить нижние и верхние (для радиоимпульса). Запишите в тетрадь граничные частоты полос пропускания и задерживания.

7. Синтезируйте первый из предложенных вам фильтров, нажав кнопку *New Design*, расположенную под списком *Filters* в окне *SPTool*: *startup.spt.* Откроется окно *Filter Designer* (рис. 52). Укажите класс фильтра в раскрывающемся списке *Algorithm*. В разделе *Specifications* задайте требования к АЧХ и укажите тип фильтра (*lowpass* – ФНЧ, *highpass* – ФВЧ, *bandpass* – ППФ, *bandstop* – ПЗФ). Вы можете задать порядок фильтра или указать, что требуется синтезировать фильтр с наименьшим порядком, характеристики которого удовлетворяли бы предъявленным к ним требованиям. Нажмите кнопку *Apply*.

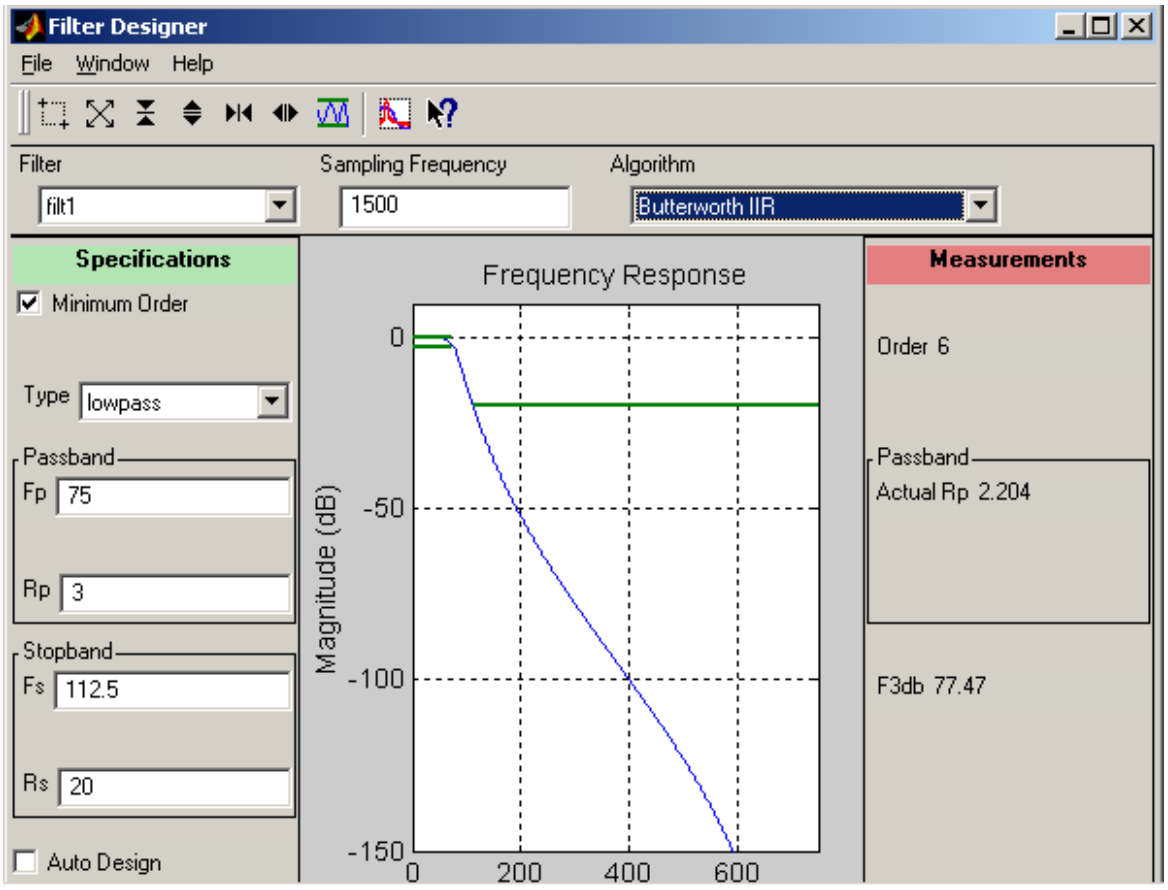


Рис. 52. Окно Filter Designer

Вернувшись в окно *SPTool: startup.spt*, выделите идентификатор синтезированного фильтра в списке *Filters* и нажмите кнопку *View*. Откроется окно *Filter Viewer*. Просмотрите характеристики фильтра. Занесите АЧХ фильтра в тетрадь, указав полосы пропускания и задерживания. (Удобно расположить этот график под графиком амплитудного спектра, изображённого ранее).

8. Осуществите фильтрацию сигнала. С этой целью выделите идентификаторы сигнала и фильтра в соответствующих списках окна *SPTool*: *startup.spt*. Нажмите кнопку *Apply*. В появившемся окне *Apply Filter* (рис. 53) задайте имя выходного сигнала в поле *Output Signal*. В раскрывающемся списке *Algorithm* задайте алгоритм фильтрации: *Direct Form II Transposed (filter)*. Затем нужно щёлкнуть по кнопке *OK*. Выходной сигнал появится в списке сигналов основного окна программы. Выделите входной и выходной сигналы в списке сигналов (при нажатии левой кнопки мыши удерживайте клавишу <*Ctrl*>).

Нажмите кнопку *View* под списком сигналов. Просмотрите графики входного и выходного сигнала и зарисуйте их.

Далее вернитесь в окно *SPTool: startup.spt*, выделите в списке сигналов выходной сигнал фильтра и нажмите кнопку *Create*. Получите спектр выходного сигнала и зарисуйте его.

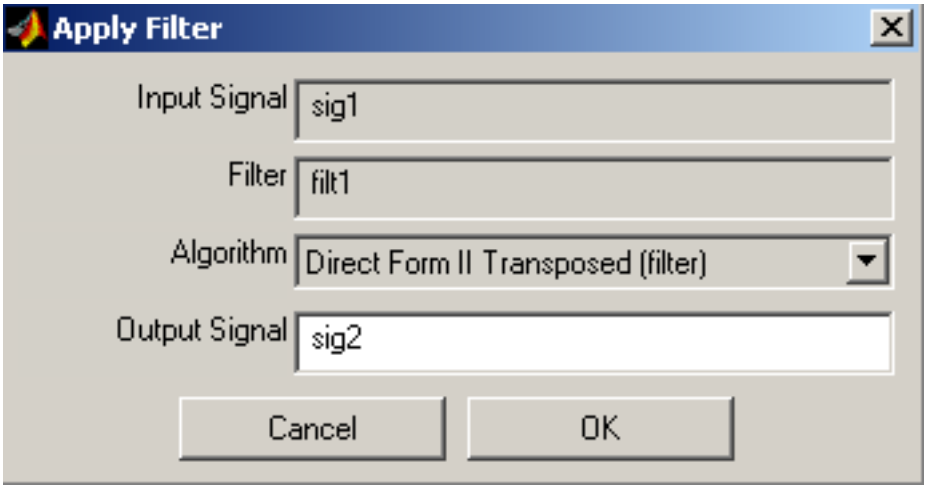


Рис. 53. Окно Apply Filter

Имеется возможность наложить спектр сигнала на график АЧХ фильтра. Для этого следует открыть окно *Filter Designer* (кнопки *New Design* или *Edit Design* основного окна *sptool*), нажать кнопку *Overlay*

Spectrum (предпоследняя на панели инструментов) и далее выбрать нужный спектр из списка в окне *Overlay Spectrum*, раздел *Select a spectrum*). В выводе отметьте, как изменился спектр сигнала и как изменилась форма сигнала.

9. Синтезируйте второй фильтр из предложенных вам в задании, профильтруйте сигнал (повторите пп. 7 и 8). Сравните результаты фильтрации первым и вторым фильтрами.

10. Измените порядок фильтра, сохранив требования только лишь в полосе пропускания. Вы можете в окне *Filter Designer* указать нужный порядок, сняв флажок в поле *Minimum Order*. Измените порядок в большую или меньшую сторону. Профильтруйте сигнал. Отметьте, как изменяются форма выходного сигнала и его спектр в зависимости от порядка фильтра.

11. Вернитесь в командное окно *MATLAB*. Измените длительность импульса. В случае, если вы фильтруете радиоимпульсы, измените частоту несущего колебания, задав небольшую расстройку между частотой несущего и центральной частотой полосового фильтра. Повторите пп. 4 – 8 для одного из предложенных вам фильтров. Отметьте в выводе, как изменяется форма профильтрованного сигнала в зависимости от его длительности и частоты расстройки.

Содержание отчета

Отчет должен содержать

1. График исходного сигнала.
2. Графики входного и выходного сигналов.

Контрольные вопросы

1. Каковы должны быть полосы пропускания и задерживания фильтра, чтобы пропустить нижние частоты спектра и подавить верхние или наоборот, или пропустить средние частоты и подавить нижние и верхние (для радиоимпульса)?

2. Определите, каковы должны быть полосы пропускания и задерживания фильтра, чтобы пропустить нижние частоты спектра и подавить верхние или наоборот, или пропустить средние частоты и подавить нижние и верхние (для радиоимпульса)?

Лабораторная работа № 9

ПРОЕКТИРОВАНИЕ ЦИФРОВОГО ФИЛЬТРА В СРЕДЕ MATLAB

Цель работы: знакомство со средой для проектирования цифровых фильтров – *fdatool* для *Matlab*. Изучение основ цифровой фильтрации. Проектирование цифровых фильтров с заданными характеристиками.

Теоретическая часть

Цифровые фильтры

Цифровые системы - это системы с цифровыми сигналами на входе и выходе. Их ядром обычно является ЦВМ. Человечество создало мало объектов, имеющих цифровую природу, поэтому общий термин «цифровая система» применяется редко. Гораздо чаще встречаются термины «цифровой фильтр» или «система цифрового управления», которые ярко отражают основную область применения этих систем. Нередко систему цифрового управления также называют цифровым фильтром. Итак, цифровой фильтр - это дискретно-временная система, выходной сигнал которой является модифицированной версией входного сигнала.

Фильтры являются основой для большинства приложений обработки сигналов. Типичное назначение - это извлечение или вырезка области спектра входного сигнала или определенной частоты. Используемые для кондиционирования сигналов фильтры нередко называются частотно-селектирующими, поскольку обычно разрабатываются на основе требований к частотной характеристике.

Понятие о разностном уравнении (РУ)

В силу дискретной природы цифровых фильтров аналоговые сигналы принимаются к обработке только в дискретные моменты времени. Информация о промежуточных значениях сигнала теряется. Таким образом, обрабатываемая цифровым фильтром входная непрерывная функция становится решетчатой.

Аналогом первой производной для решетчатой функции считается обратная разность $\nabla f[n] = f[n] - f[n-1]$.

Аналогом второй - вторая обратная разность $\nabla^2 f[n] = \nabla f[n] - \nabla f[n-1] = f[n] - 2 f[n-1] + f[n-2]$.

Аналогом дискретного уравнения (ДУ) для цифрового фильтра является уравнение в конечных разностях, или разностное уравнение. Как и непрерывные системы, цифровые фильтры могут быть описаны совокупностью РУ или одним, решенным относительно требуемой координаты. В общем случае цифровой фильтр, имеющий один вход и один выход, описывается РУ:

$$y[k] = \left(\sum_{i=0}^{m} b_i x[k-i] - \sum_{j=1}^{n} a_j y[k-j] \right) / a_0$$

где $y[k]$ - выходная координата цифрового фильтра; $x[k]$ – входная координата; a_j и b_i - постоянные коэффициенты, они же фигурируют в знаменателе и числителе соответствующей дискретной передаточной функции (z-ПФ).

Это уравнение отражает взаимосвязь между k-м выходным значением и рядом предыдущих значений на входе и выходе в количестве m и n соответственно.

Если коэффициенты a_j равны нулю ($a_0 = 0$), то РУ и программа, его реализующая, нерекурсивные; фильтр обладает конечной импульсной характеристикой и называется КИХ-фильтром.

Если коэффициенты a_j не равны нулю, то РУ и программа, его реализующая, рекурсивные; фильтр обладает бесконечной импульсной характеристикой и называется БИХ-фильтром.

Итак, выходное значение цифрового фильтра – есть взвешенная сумма текущего и нескольких предыдущих значений как входного сигнала, так (в случае БИХ-фильтров) и предыдущих значений выходного сигнала.

При выборе фильтра Бесселя расчет коэффициентов полиномов ведется из стремления аппроксимировать ЛФЧХ фильтра (рис. 54) таким образом, чтобы запаздывание на всех частотах было одинаковым, фильтр Бесселя не искажает сигнал, спектр которого лежит в полосе пропускания. Вследствие чего переходная характеристика фильтра

имеет очень малое перерегулирование. ЛАЧХ этого фильтра не колеблется ни в полосе пропускания, ни в полосе подавления (рис. 55).

Рис. 54. График логарифмической фазочастотной характеристики фильтра Бесселя

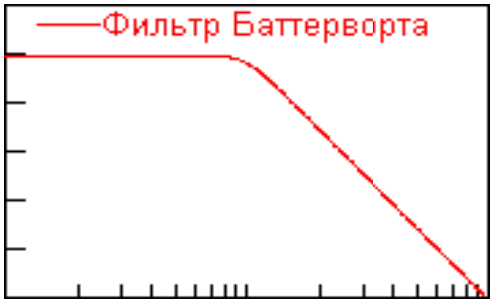


Рис. 55. График логарифмической амплитудно-частотной характеристики фильтра Баттерворта

Фильтр Баттерворта (*Butterworth*)

При выборе фильтра Баттерворта расчет коэффициентов полиномов ведется из стремления аппроксимировать максимально плоскую АЧХ фильтра, которая при нормировании описывается функцией $|W(j\omega)| = 1 / (\omega^{2n}-1)^{1/2}$, где: $\omega = \omega / \omega_{ср}$ - относительная частота; $\omega_{ср}$ - частота среза; n - порядок фильтра.

Все производные аппроксимирующей функции по частоте от первой до $(2n-1)$-й включительно в точке $\omega = 0$ равны нулю (т.е. АЧХ плоская).

Фильтры Чебышева и инверсный Чебышева (*Chebyshev & Inverse Chebyshev*) (рис. 56, 57).

При выборе фильтра Чебышева расчет коэффициентов полиномов ведется из стремления аппроксимировать АЧХ фильтра с максимальным подавлением. Ценой будет неравномерность АЧХ в полосе пропускания, не превышающая за-

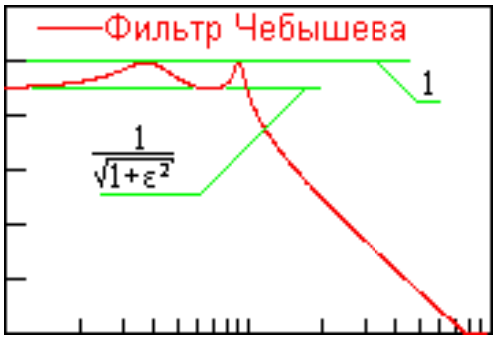


Рис. 56. График логарифмической амплитудно-частотной характеристики фильтра Чебышева

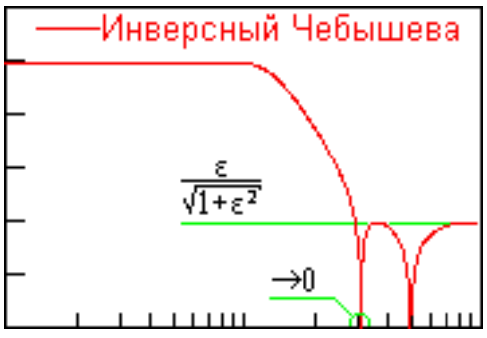


Рис. 57. График логарифмической амплитудно-частотной характеристики инверсного фильтра Чебышева

данной величины. Наилучшая аппроксимация описывается функцией $|W(j\omega)|^2 = 1 / (1+\varepsilon^2 T_n^2(\omega))$, где ε - постоянный коэффициент, определяющий неравномерность АЧХ фильтра; T_n - полином Чебышева первого рода *n*-го порядка.

В полосе пропускания квадрат АЧХ $|W(j\omega)|^2$ фильтра колеблется между уровнями, равными 1 и $1 / (1+\varepsilon^2)$, причем число таких колебаний ("волн" на графике АЧХ) тем больше, чем выше порядок фильтра. Амплитуда этих колебаний одинакова.

В инверсном фильтре Чебышева АЧХ монотонно изменяется в полосе пропускания и пульсирует в полосе заграждения. Аппроксимация АЧХ описывается функцией

$|W(j\omega)|^2 = \varepsilon^2 T_n^2(1/\omega) / (1+\varepsilon^2 T_n^2(1/\omega))$.

В полосе заграждения квадрат АЧХ $|W(j\omega)|^2$ фильтра колеблется между значениями 0 и $\varepsilon^2 / (1+\varepsilon^2)$.

Главное окно программы показано на рис. 58, оно появляется на экране после загрузки программы, для чего надо набрать её имя *>> fdatool*.

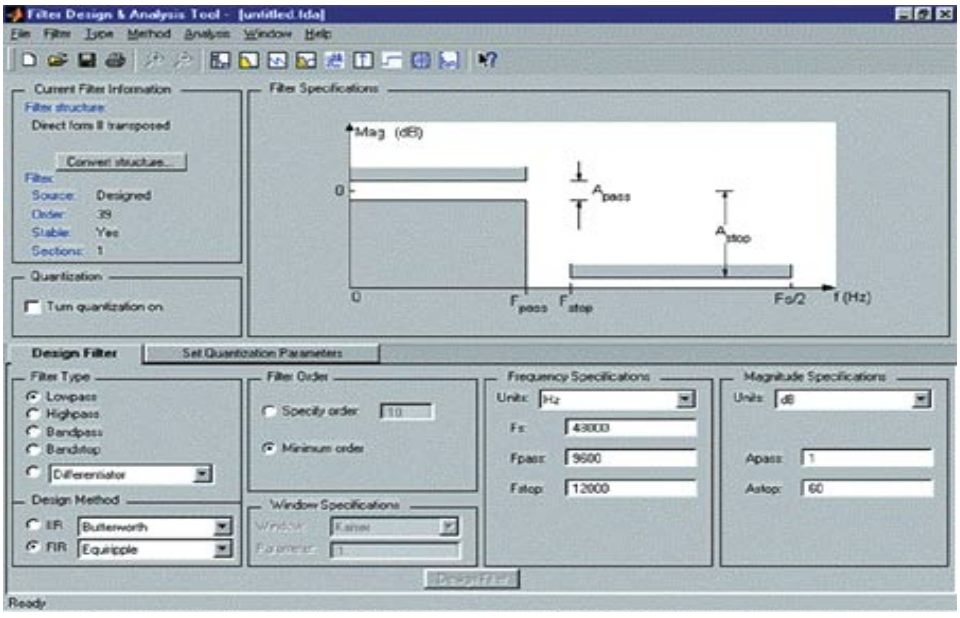


Рис. 58. Главное окно программы *fdatool*

Как видно из рисунка, главное окно включает несколько областей, или разделов, для ввода и вывода необходимой информации. В левом

верхнем углу находится раздел Current Filter Information (информация о фильтре), где отображается информация о фильтре, с которым в текущий момент выполняется работа, а именно:

- форма реализации;
- источник (получен ли фильтр с помощью *fdatool* или импортирован из другого приложения);
- устойчивость;
- количество звеньев.

В дополнение к перечисленному в этом разделе находится кнопка *Convert Structure* (преобразовать структуру), с помощью которой можно выбирать форму (схему) реализации фильтра.

Правее раздела *Current Filter Information* находится область, предназначенная для графического и численного отображений следующих характеристик и параметров фильтров:

- *Filter Specifications* (спецификация);
- *Magnitude Response* (амплитудно-частотная характеристика, или АЧХ);
- *Phase Response* (фазочастотная характеристика, или ФЧХ);
- *Magnitude and Phase Response* (АЧХ и ФЧХ);
- *Group Delay* (групповое время задержки);
- *Impulse Response* (импульсная характеристика);
- *Step Response* (переходная характеристика);
- *Pole/Zero Plot* (полюсы и нули);
- *Filter Coefficients* (коэффициенты фильтра).

Сразу после загрузки программы в этой области, как показано на рис. 58, отображается окно спецификации фильтра. Для отображения того или иного окна нужно нажать на одну из кнопок, расположенных вверху главного окна под строкой меню. Эти кнопки, а также их назначение показаны на рис. 59 в виде увеличенного фрагмента главного окна. Кроме отмеченных кнопок, те же действия можно выполнить, выбирая соответствующие пункты меню *Analysis*.

Нижняя половина главного окна содержит две страницы: Design Filter (проектирование фильтра) и *Set Quantization Parameters* (установка параметров дискретизации), причём сразу после загрузки про-

граммы активна страница *Design Filter* (см. рис. 58), заполнение которой позволяет рассчитывать фильтры без учёта эффектов квантования, то есть с машинной точностью.

Рис. 59. Управляющие кнопки для отображения характеристик и параметров фильтров

В левой части этой страницы располагаются разделы *Filter Type* (выбор типа фильтра) и *Design Method* (метод проектирования). Выбрав соответствующую строку раздела *Filter Type*, пользователь тем самым выбирает фильтр, который собирается проектировать. Выбор включает следующие типы фильтров:

- *Lowpass* (нижних частот);
- *Highpass* (верхних частот);
- *Bandpass* (полоснопропускающий);
- *Bandstop* (полоснозаграждающий),

а также следующие типы специализированных цифровых цепей:

- *Differentiator* (дифференциатор);
- *Hilbert Transformer* (преобразователь Гильберта);
- *Multiband* (многополосный фильтр);
- *Arbitrary Magnitude* (фильтр с произвольной АЧХ, форма которой определяется пользователем);
- *Arbitrary Group Delay* (фильтр с произвольным групповым временем задержки).

Раздел *Design Method* позволяет сделать выбор между фильтрами с бесконечными и конечными импульсными характеристиками (*IIR - Infinite Impulse Response* и *FIR - Finite Impulse Response* соответственно). В случае выбора IIR-фильтров необходимо конкретизировать вид проектируемого фильтра: Баттерворта, Чебышева 1-го рода, Чебышева 2-го рода и эллиптический (рис. 60, *а*). Если же выбраны *FIR*-фильтры, выбор включает расчёт равноволновых фильтров, а также

расчёт методом наименьших квадратов и оконным методом (рис. 60, *б*). Если проектирование выполняется оконным методом, активизируется область *Window Specifications* (спецификация окна), расположенная рядом с областью *Design Method*, в которой имеется раскрывающееся меню *Window* с набором спектральных окон.

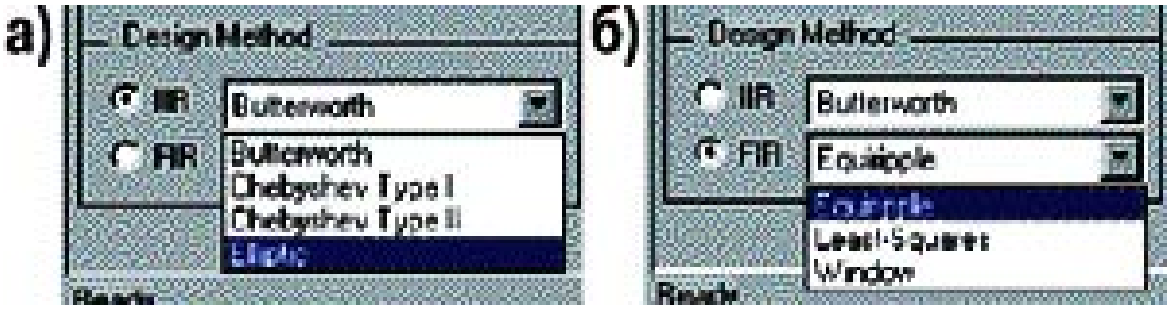


Рис. 60. Выбор метода расчёта при проектировании фильтров с бесконечными (а) и конечными (б) импульсными характеристиками

Правую часть страницы *Filter Design* занимают разделы *Frequency Specifications* и *Magnitude Specifications* (частотные и амплитудные спецификации соответственно). Первый из них содержит редактируемые окна для ввода значений частоты дискретизации *Fs* и граничных частот полос пропускания *Fpass* и задерживания *Fstop*, а также единиц измерения АЧХ - *Units*, причём количество граничных частот зависит от типа фильтра, задаваемого в разделе *Filter Type*. Второй раздел - *Magnitude Specifications* - позволяет задать ограничения амплитудной характеристики проектируемого фильтра для областей пропускания (*Apass* или *Wpass*) и задерживания (*Astop* или *Wstop*). Оба обсуждаемых раздела имеют раскрывающиеся меню *Units* для выбора единиц измерения частоты и амплитуды. На странице *Filter Design* имеется ещё одна область, а именно *Filter Order* (порядок фильтра) в которой можно указать явно порядок проектируемого фильтра либо потребовать, чтобы программа автоматически выбрала наименьший порядок в соответствии с введённой спецификацией.

Порядок выполнения работы

Придерживаясь терминологии, используемой в справочной системе *MATLAB*, цифровой фильтр, коэффициенты которого рассчитаны с машинной точностью, будем называть цифровым фильтром-прототипом (в оригинале - *Referenced Filter*).

1. Выделить и заполнить соответствующие позиции на странице *Filter Design* главного окна *fdatool*, как показано на рис. 61 - 64.

2. Нажать на клавишу *Design Filter*, расположенную внизу окна. После завершения расчёта в верхней части окна будут представлены результаты.

3. Провести расчет с заданными параметрами четырех фильтров Баттервора и Чебышева 1-2-го порядков. Зарисовать полученную АЧХ.

4.

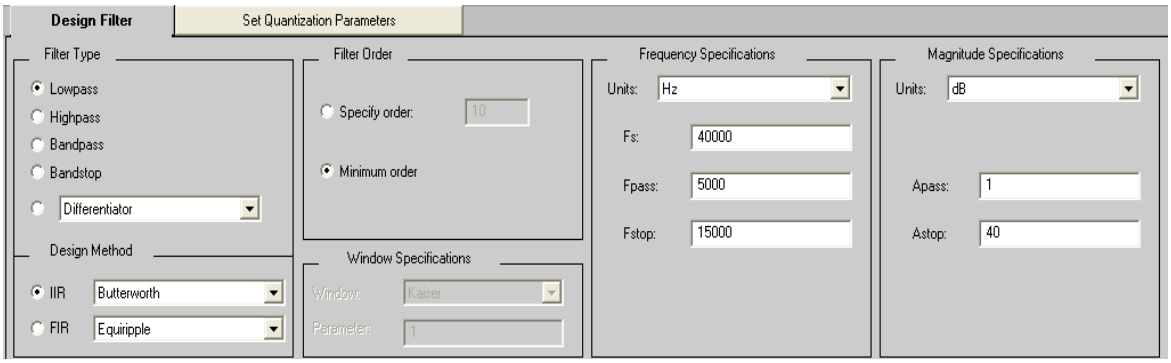


Рис.61. Окно задания параметров фильтра нижних частот

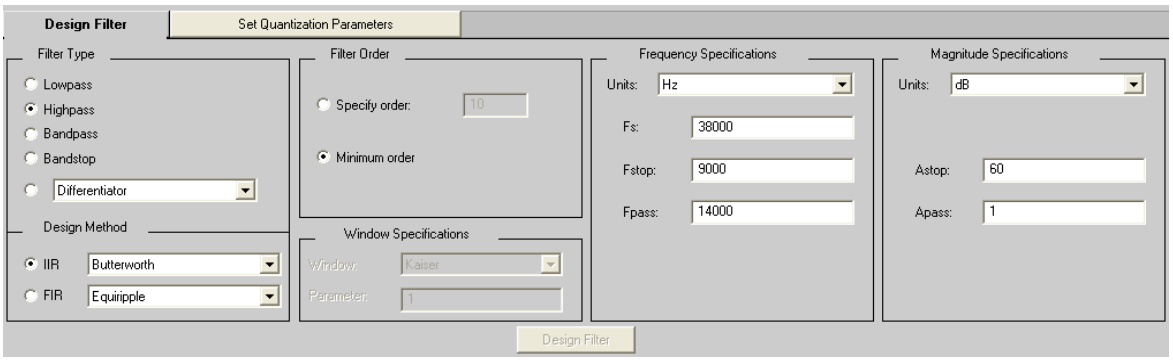


Рис. 62. Окно задания параметров фильтра верхних частот

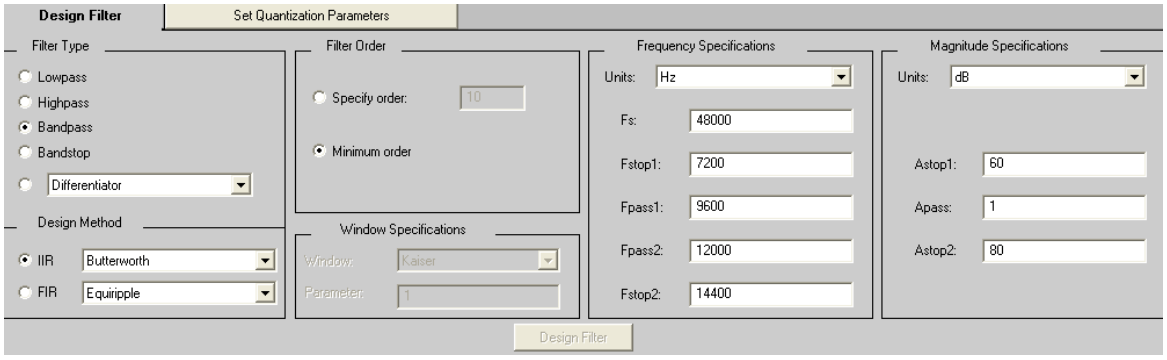


Рис. 63. Окно задания параметров полоснопропускающего фильтра

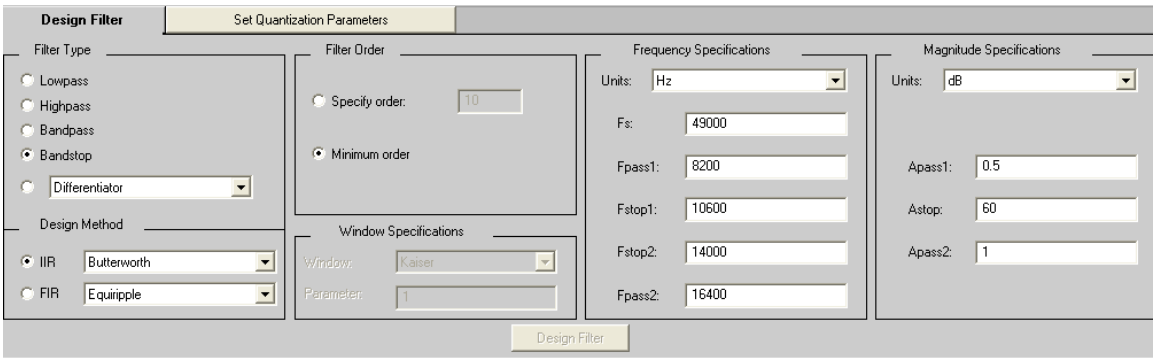


Рис. 64. Окно задания параметров полоснозаграждающего фильтра

В области отображения характеристик и параметров выводится АЧХ полученного фильтра, однако можно по мере необходимости для вывода переключать её содержимое, например, импульсной характеристики, полюсов, нулей и так далее.

Контрольные вопросы

1. Что такое цифровой фильтр?
2. Какие фильтры лучше использовать для обработки получаемых сигналов?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Основной

1. Темников, Ф. Е. Теоретические основы информационной техники / Ф. Е. Темников, В. А. Афонин, В. Н. Дмитриев. – М.: Энергия, 1971. – 424 с.

2. Макс, Ж. Методы и техника обработки сигналов при физических измерениях: В 2 т. Т. 1. / Ж. Макс; пер. с фр. – М.: Мир, 1983. – 312 с.

3. Макс, Ж. Методы и техника обработки сигналов при физических измерениях: В 2 т. Т. 2. / Ж. Макс; пер. с фр. – М.: Мир, 1983. –256 с.

Дополнительный

1. Солодов, А. В. Теория информации и ее применение к задачам автоматического управления и контроля / А. В. Солодов. – М.: Нация, 1967. – 384 с.

2. Зиновьев, А. Л. Введение в теорию сигналов и цепей / А. Л. Зиновьев, Л. Н. Филиппов. – М.: Высш. шк., 1975. – 264 с.

3. Хлистунов, В. Н. Основы цифровой электроизмерительной техники / В. Н. Хлистунов. – М.: Энергия, 1966. – 345 с.

ОГЛАВЛЕНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ
ПО ДИСЦИПЛИНЕ «ПРЕОБРАЗОВАНИЕ ИЗМЕРИТЕЛЬНЫХ СИГНАЛОВ»

Составитель

ЛЕГАЕВ Владимир Павлович

Подписано в печать 27.04.09.

Формат 60x84/16. Усл. печ. л. 5,11. Тираж 100 экз.

Заказ

Издательство

Владимирского государственного университета.

600000, Владимир, ул. Горького, 87.